Události

V současném světě Internetu musíme vyvíjet webové aplikace dynamicky a interaktivně. Musíme tudíž reagovat na vstup uživatele tak, abychom mu přinesli co největší komfort. Programovací jazyk JavaScript nabízí několik možností pro interakci s uživatelem. Psaní kódu v tomto jazyku ale není tak elegantní jako v knihovně jQuery, která zjednodušuje a rozšiřuje základní zpracování událostí. V této kapitole se dozvíme, co to události jsou a jak je efektivně používat ve svých aplikacích. Postupně si důkladně popíšeme zpracování základních událostí, a to takových, které vznikají jak při práci s myší, tak při práci s klávesnicí. Zpracování některých událostí si předvedeme na jednoduchých příkladech, abychom získali lepší představu o tom, jak události pracují.

Co jsou události

Události jsou velmi důležitou součástí každé webové aplikace nebo webové stránky. V začátcích Internetu se webové stránky skládaly jen ze statického textu a události se na nich vůbec nezpracovávaly. Technika tvorby webových stránek ale výrazně pokročila kupředu a my můžeme vyvíjet aplikace plné dynamiky, k čemuž nám slouží hlavně události.

Událost je akce, která vzniká na stránce. Může jí být kupříkladu pohyb kurzorem myši, stisknutí klávesy na klávesnici anebo klepnutí na libovolný prvek umístěný na stránce. Mezi pěkné příklady patří dynamické formuláře, navigační nabídky, zpracování map společnosti Google, různé interaktivní fotogalerie s možností editace apod. Na tyto události musíme nějakým způsobem zareagovat a následně je zpracovat. V knihovně jQuery je možné zpracovat událost velmi jednoduše, jak je patrné z výpisu 5.1.

Výpis 5.1. Zpracování události v knihovně jQuery

```
$(document).ready(function () {
   $(element).udalost(function () {
      // akce knihovny jQuery
   });
});
```

Objekt zapsaný do jednoduchých závorek za znakem dolaru představuje vybraný element, jehož událost chceme ošetřit. Za tímto elementem následuje název metody, s níž zpracováváme nezbytnou událost. Před název této metody vkládáme tečku. Nakonec zapíšeme mezi složené závorky anonymní funkce `function() {}` akci, která se má vykonat při vzniku této události.

Načtení dokumentu

Úplně první událostí, kterou bychom měli umět zpracovat, je ta, která vzniká při načtení dokumentu. V tomto případě se ale nemusíme učit nic nového, protože jsme se s ní seznámili už ve třetí kapitole. Připomeňme si, že událost načtení stránky zpracováváme v kódu `$(document).ready()`. O obsluhu události načtení dokumentu se tedy stará metoda `ready()` knihovny jQuery. Tato metoda se spouští hned po kompletním načtení modelu DOM, přičemž částečně nahrazuje funkci `load()`, kterou známe z jazyka JavaScript. Metodu `ready()` a její použití jsme si podrobně popsali v minulé kapitole. Zde si pouze řekneme, že ji lze kromě běžného zápisu z výpisu 5.2. zapsat také zkráceně.

Výpis 5.2. Standardní zápis metody ready()

```
$(document).ready(function () {
   // akce knihovny jQuery
});
```

Výpis 5.3. Zkrácený zápis metody ready()

```
$().ready(function () {
   // akce knihovny jQuery
});
```

Používáme události

Zpracováním událostí výrazně zlepšujeme interaktivitu svých webových aplikací, bez níž se současná webová aplikace neobejde. Bez zpracování událostí by byly naše stránky pro uživatele velmi nepohodlné. Na co všechno můžeme tedy události používat?

Můžeme kupříkladu zvýraznit tlačítko, jakmile nad něj uživatel přesune ukazatel myši. Rovněž můžeme reagovat na to, když uživatel píše do textového pole. Existuje spousta možností jak a na co používat události. Pro začátek si ukážeme, jak funguje jednoduchá událost `click`. V uvedeném výpise si zvýrazníme část kódu, která se vykoná po zachycení události. Samotnou událost si samozřejmě popíšeme detailněji za malou chvíli.

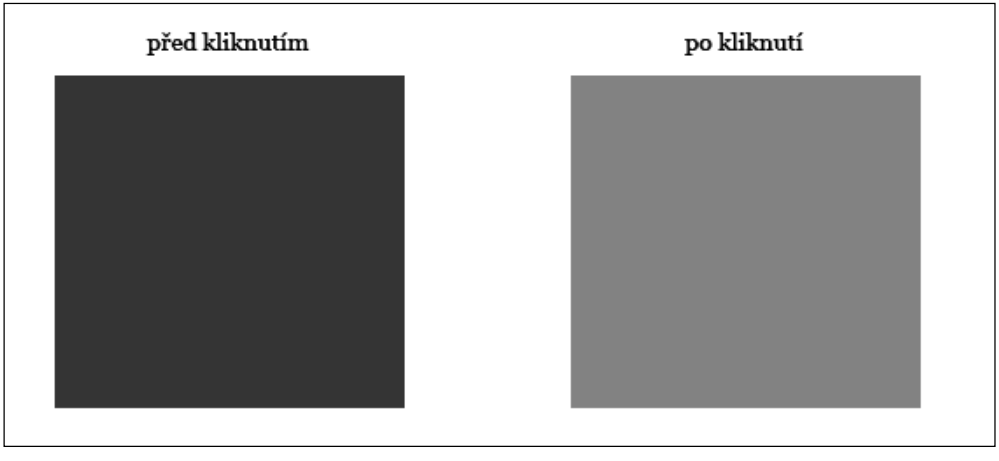
Poznámka: Názvy událostí v knihovně jQuery se podobají názvům událostí v jazyce JavaScript. V případě knihovny jQuery ale nezačínají názvy událostí předponou `on`. Kupříkladu událost `onclick` z jazyka JavaScript se v knihovně jQuery jmenuje pouze `click`.

Jak už jsme si řekli, obsluhující funkci pro událost zapisujeme tak, že nejprve vybereme příslušný element a potom na něj zavoláme příslušnou metodu. Výpis 5.4 obsahuje zdrojový kód napsaný v knihovně jQuery, v němž zpracováváme událost klepnutí myši na vybraný element. V tomto případě se jedná o element s identifikátorem `pokus`.

Výpis 5.4. Událost klepnutí myší

```
$(document).ready(function() {
   $("#pokus").click(function(){
      $("#pokus").css("background","red");
   });
});
```

Jestliže klepneme na element s identifikátorem `pokus`, vznikne událost `click` a poté se spustí metoda `click()`, v níž se nachází náš obslužný kód. V tomto obslužném kódu měníme barvu pozadí daného elementu na červenou. Výsledek můžete vidět na obrázku 5.1.



Obrázek 5.1. Stav před klepnutím a po klepnutí na element s identifikátorem pokus

Základní události

Popíšeme si události, které vznikají v prohlížeči a události, s nimiž se setkáme při práci s formuláři jazyka HTML. Rozdílné webové prohlížeče nakládají s událostmi jiným způsobem. Proto je problematické zpracovat události tak, aby nám výsledný kód fungoval ve všech webových prohlížečích. I v tomto případě nám knihovna jQuery zabezpečí kompatibilitu a my se nebudeme muset trápit s psaním zdrojového kódu pro různé prohlížeče. Ukážeme si jednoduchý příklad plovoucího objektu. Vytvoříme blok, který se bude posouvat, zatímco uživatel se bude posouvat ve stránce. Nejprve si musíme nachystat samotný blok, který posléze s pomocí knihovny jQuery oživíme. Nachystaný blok najdete ve výpisu 5.5.

Výpis 5.5. Plovoucí blok

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
   <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
   <title>Události - Plovoucí blok</title>
   <link rel="stylesheet" href="styl.css" type="text/css" />

   <script src="js/jquery.js" type="text/javascript"></script>
   <script src="js/priklad_5.1.js" type="text/javascript"></script>
</head>
<body>
   <div id="hlavni">
      <div id="top"><h1>Události - Plovoucí blok</h1></div>
      <div id="blok"></div>

      <div id="obsah">
         <h1>Obsah stránky</h1>
      </div>
   </div>
</body>
</html>
```

Námi vytvořená stránka obsahuje provizorní záhlaví a prostou obsahovou část. Kvůli zachování jednoduchosti jsme nevytvářeli propracovaný grafický návrh, ale jednoduchou ohraničenou část stránky. Napíšeme tedy kaskádové styly, díky nimž budeme moct určit rozměry a vzhled jednotlivých elementů. Definujeme pravidlo pro hlavní blok, který jsme pojmenovali `hlavni`. Tomuto bloku nastavíme šířku 850 pixelů. Dále napíšeme pravidla pro bloky s identifikátory `top` a `obsah` nacházející se v hlavním

bloku. Pravidlem pro element s identifikátorem `top` definujeme vzhled provizorního záhlaví tak, že mu nastavíme šířku 50 pixelů a ohraničíme jej šedým rámečkem. Element s identifikátorem `obsah` představuje obsahovou část naší stránky. Tomuto elementu nastavíme výšku 1 500 pixelů a rovněž jej ohraničíme šedým rámečkem. Ještě nám zbývá samotný plovoucí blok s identifikátorem `blok` – tomu nastavíme šířku i výšku na 150 pixelů a černý rámeček. Kromě toho mu nastavíme absolutní polohu, odsadíme jej shora o 50 pixelů a zarovnáme jej na pravou část naší stránky. Všechna tato pravidla najdete ve výpise 5.6.

Výpis 5.6. Doplněná pravidla jazyka CSS

```
#hlavni {
   width: 850px;
   margin: 0 auto;
}

#top {
   border: 1px solid #c3c3c3;
   margin: 5px;
   height: 50px;
}

#obsah {
   border: 1px solid #c3c3c3;
   margin: 5px;
   height: 1500px;
}

#blok {
   width: 150px;
   height: 150px;
   float: right;
   border: 1px solid #000000;
   position: absolute;
   margin-left: 80%;
   top: 50px;
   z-index: 1;
}
```

Obsluha událostí

V následujícím obslužném kódu napsaném v knihovně jQuery oživíme náš dosud nehybný blok. Abyste získali lepší představu o tom, jak knihovna jQuery zachytává a zpracovává události, tento blok bude obsahovat také informace o aktuální pozici v posouvané stránce. Prohlédněte si tedy výpis 5.7 a potom se dozvíte, jak funguje.

Výpis 5.7. Obsluha událostí prohlížeče v knihovně jQuery

```
$(document).ready(function($) {
   var body = parseInt($('#obsah').offset().top) - 20;

   $(window).scroll(function () {
      var scroll = $(window).scrollTop();

      $('#blok').html("posunutí: " + scroll);
      if (scroll > body) {
         $('#blok').stop().css({
            position: 'fixed',
            left: 407,
            top: 20
         });
      } else if (scroll < body) {
         $('#blok').css({
            left: 407,
            top: 50,
         });
      }
   });
});
```

Díky knihovně jQuery jsme oživili náš statický blok. Po spuštění tohoto příkladu můžeme pozorovat, jak se náš blok přemisťuje, když posouváme danou stránku. V praxi můžeme tento příklad použít ke zviditelnění důležité informace. V kódu jsme nejprve deklarovali proměnnou `body`, jejíž hodnotu jsme získali z metody `offset()` knihovny jQuery. Z objektu, jejž tato metoda vrací, jsme totiž načetli vlastnost `top`. Proměnná `body` tudíž obsahuje svislou pozici elementu s identifikátorem `obsah`.

Poznámka: Prostřednictvím metody `offset()` určujeme aktuální polohu elementu. Tato metoda vrací objekt s vlastnostmi `top` a `left`. Jestliže potřebujeme pouze jednu z těchto vlastností (například vlastnost `left`), použijeme kód `.offset().left`.

Posléze zpracováváme událost `scroll` pro element `window`, který reprezentuje okno našeho webového prohlížeče. Událost `scroll` vzniká a zpracovává se, zatímco uživatel posouvá stránku. Do složených závorek obslužné funkce pro tuto událost jsme zapsali kód, který se vykoná při každému posunutí stránky. V tomto kódu máme mimo jiné proměnnou `scroll`, jejíž hodnotu získáváme voláním metody `scrollTop()` na element `window`.

Poznámka: Metoda `scrollTop()` vrací aktuální svislou pozici posuvníku pro daný element.

Takto získanou hodnotu následně vkládáme do elementu s identifikátorem `blok`, a proto v něm po spuštění tohoto příkladu uvidíme aktuální pozici posouvaného okna. Nakonec podle platnosti jednoduché podmínky nastavujeme vzhled elementu s identifikátorem `blok`. Díky tomu se náš blok neustále posouvá při posouvání stránky, a tak ho máme stále na očích.

Obrázek 5.2. Plovoucí blok v akci

Jaké události lze zachytávat

Událost `scroll` samozřejmě není jediná – v knihovně jQuery můžeme zachytávat spoustu dalších událostí. Zachytávání a zpracování ostatních událostí prohlížeče a formuláře je v podstatě úplně stejné. Podrobný popis všech událostí tohoto typu by bylo na rámec této knihy, proto si popíšeme alespoň ty nejdůležitější z nich. Následně si ukážeme zajímavé příklady zpracování událostí myši a klávesnice.

- Událost `error` – tato událost vzniká v případě, že se daný element nenačetl správně.
- Událost `resize` – tato událost nastává, když se změní velikost okna prohlížeče.
- Událost `focus` – tato událost vzniká, když uživatel přesune zaměření na daný element. Můžeme ji aplikovat pouze na některé elementy.
- Událost `blur` – tato událost je opakem události `focus` – nastává, když element ztratí zaměření.
- Událost `change` – tato událost vzniká v případě, že se změní hodnota elementu.
- Událost `select` – tato událost nastává, když uživatel označí jakýkoliv text; je však omezená na element `input` typu `text` a na element `textarea`.
- Událost `submit` – tato událost vzniká, jakmile se uživatel pokusí odeslat formulář.

Událost vznikající při špatném načtení elementu

Způsob použití výše uvedených základních událostí si předvedeme na krátkých příkladech. Začneme událostí `error`, jejíž použití se nachází na výpise 5.9. Jak už víme, tato událost nastává v případě, že se daný element nenačte správně. Ukážeme si ji na příkladě, v němž chceme načíst obrázek. Zápis obrázku vypadá v běžném dokumentu HTML jako ve výpise 5.8.

Výpis 5.8. Zápis obrázku v standardním dokumentu HTML

```
<img id="test" alt="testovací obrázek" />
```

Jak jste si jistě všimli, kód z výpisu 5.8 neobsahuje adresu, na níž se daný obrázek nachází. Tuto adresu obrázku vkládáme běžně do atributu `src`, který ale v tomto případě schází. Je tedy nadmíru jasné, že se žádný obrázek nenačte. Jinými slovy – načítání obrázku vyvolá událost `error`, kterou ošetříme ve výpisu 5.9. Po zachycení této události se na obrazovku vypíše text „Obrázek se nenačetl správně." Posléze vložíme do atributu `src` obrázek `chyba.jpg`, jenž už se zobrazí správně. Tímto způsobem můžeme nahradit kupříkladu obrázky, které uživatel nahrál špatně na náš server.

Výpis 5.9. Použití události error v knihovně jQuery

```
$(document).ready(function() {
   $("#test").error(function() {
      alert("Obrázek se nenačetl správně.");
   })
   .attr("src", "chyba.jpg");
});
```

Událost při změně hodnoty elementu

Jako další si představíme událost `change` a rovněž si ji předvedeme na krátkém příkladu. Tato událost vzniká, pokud změníme hodnotu daného elementu. Může nastat pouze u některých elementů – konkrétně u elementů `textarea`, `input` a `select`. Ukážeme si proto příklad s jednoduchým rozevíracím seznamem, pro nějž zpracujeme událost `change`. Takto připravený seznam se nachází ve výpise 5.10.

Výpis 5.10. Jednoduchý seznam jazyka HTML

```
<select name="lokalita">
   <option>Kralický sněžník</option>
   <option>Slávkovský les</option>
   <option>České středohoří</option>
   <option>Novohradské hory</option>
   <option>Jizerské hory</option>
</select>
```

Následně zpracujeme událost, která vznikne, kdykoliv vybereme novou hodnotu z našeho seznamu. Ve výpisu 5.11 se můžete podívat, jak si s touto událostí poradí knihovna jQuery. Náš příklad je velmi jednoduchý – jediné, co děláme po zachycení dané události, je to, že vypíšeme vybranou hodnotu na obrazovku.

Výpis 5.11. Použití události change v knihovně jQuery

```
$(document).ready(function() {
   $("select").change(function() {
      alert("Vybrali jste hodnotu: " + this.value)
   });
});
```

Události objektu

Jelikož se seznamujeme se základními událostmi, které knihovna jQuery zpracovává, je vhodné, abychom si alespoň okrajově popsali jednotlivé události objektu. S nimi jsme se už setkali v prvním příkladu z této kapitoly, v němž jsme vytvářeli plovoucí blok. V další části této kapitoly si tyto události objektu rámcově popíšeme. Nejprve si vypíšeme vlastnosti, které můžeme použít při práci s událostmi objektu, a potom si ukážeme jednoduchý příklad, v němž je použijeme.

- Vlastnost `event.type` – tato vlastnost obsahuje jméno zpracovávané události.
- Vlastnost `event.target` – tato vlastnost obsahuje element, pro nějž zpracováváme danou událost.

- Vlastnost `event.relatedTarget` – tato vlastnost obsahuje předchozí element zpracovaný pro danou událost.
- Vlastnost `event.pageX` – tato vlastnost obsahuje aktuální souřadnici ukazatele myši na ose X vzhledem ke klientské oblasti okna prohlížeče.
- Vlastnost `event.pageY` – tato vlastnost obsahuje aktuální souřadnici ukazatele myši na ose Y vzhledem ke klientské oblasti okna prohlížeče.
- Vlastnost `event.screenX` – tato vlastnost obsahuje souřadnici ukazatele myši na ose X vzhledem k obrazovce.
- Vlastnost `event.screenY` – tato vlastnost obsahuje souřadnici ukazatele myši na ose Y vzhledem k obrazovce.
- Vlastnost `event.result` – tato vlastnost obsahuje poslední hodnotu, kterou jsme vrátili ve funkci pro zpracování dané události.

Jak používat vlastnost `event.relatedTarget` si ukážeme na jednoduchém příkladu, v němž použijeme několik různých elementů, pro které budeme zpracovávat událost `mouseout`. V tomto příkladu rovněž vypíšeme souřadnice ukazatele myši na osách X a Y vzhledem k obrazovce. Tyto souřadnice zjistíme z vlastností `event.screenX` a `event.screenY`. Daný příklad najdete ve výpise 5.12.

Výpis 5.12. Různé elementy vytvořené v jazyce HTML

```
<div style="border: 1px solid #000; width: 300px;">
   <h1>Libovolný nadpis</h1>
   <h2>Libovolný podnadpis</h2>
   <p>Jakýkoliv text na stránce</p>
   <a href="#">Odkaz na stránce</a>
</div>
<p id="souradnice">Zde budeme zobrazovat souřadnice.</p>
```

Pro jednoduchost jsme pravidlo stylu, s nímž upravujeme vzhled uvedeného elementu `div`, vložili přímo do zápisu tohoto elementu. V knihovně jQuery napíšeme kód, v němž budeme zpracovávat událost `mouseout` pro tento element `div`. V reakci na událost `mouseout` vypíšeme aktuální hodnotu vlastnosti `event.relatedTarget` na obrazovku. Všimněte si, že na obrazovce se vždy objeví předcházející element, a ne ten, přes nějž jste právě přesunuli ukazatel myši.

Současně zpracováváme událost `mousemove` pro element `document`, přičemž načítáme hodnoty vlastností daného objektu. Z těchto hodnot nás zajímají souřadnice ukazatele myši na obrazovce, které ukládáme do předem vytvořeného elementu `p` s identifikátorem `souradnice`. Tento příklad najdete na výpise 5.13.

Výpis 5.13. Použití vlastností event.relatedTarget, event.screenX a event.screenY

```
$(document).ready(function() {
   $("div").mouseout(function(event) {
      alert(event.relatedTarget.nodeName);
   });

   $(document).bind('mousemove',function(e) {
      $("p#souradnice").text("souřadnice X: " + e.screenX + ", souřadnice
Y: " + e.screenY);
   });
});
```

Události myši

Události myši patří k nejčastěji používaným událostem. Ošetřují každou operaci, kterou vykonáme na stránce s pomocí myši. Mezi tyto operace patří kupříkladu přesun ukazatele myši, klepnutí na element anebo přesunutí ukazatele myši nad element. Knihovna jQuery pracuje s několika událostmi, které reagují na akce s myší. K těmto událostem se rovněž řadí události `mouseenter` a `mouseover`, které si lidé často pletou. Vysvětlíme si, jak se od sebe liší. Obě události můžeme připojit k jakémukoliv elementu. Například událost `mouseenter` připojíme k elementu s identifikátorem `pokus` tak, jak je patrné z výpisu 5.14.

Výpis 5.14. Zpracování události mouseenter

```
$("#pokus").mouseenter(function(){
    alert("Byla zachycena událost mouseenter");
});
```

Obě události nastávají vždy, když přesuneme ukazatel myši nad daný element. Událost `mouseenter` však vzniká pouze tehdy, když přesouváme ukazatel myši z vnějšího elementu, zatímco událost `mouseover` vzniká i v případě, že přesuneme ukazatel myši z vnořeného elementu. Proto nám chování události `mouseover` může někdy způsobovat velmi vážné problémy.

Poznámka: Samozřejmě existují i události, které vznikají, když opustíme ukazatelem myši daný element. V případě události `mouseenter` se jedná o událost `mouseleave` a v případě události `mouseover` o událost `mouseout`.

Knihovna jQuery nabízí také metodu `hover()`, která zahrnuje jak událost `mouseenter`, tak událost `mouseleave`. Díky ní je zpracování událostí `mouseenter` a `mouseleave` mnohem jednodušší.

Příklad události myši

Ukážeme si příklad rozevírací nabídky, na němž si popíšeme zpracování výše uvedených událostí prostřednictvím metody `hover()`. Nejprve si vytvoříme seznam, kterému pomocí kaskádových stylů změníme vzhled. Posléze napíšeme zdrojový kód v knihovně jQuery, ve kterém ošetříme vzniklé události myši. Tak získá naše rozevírací nabídka potřebnou funkčnost. Výpis 5.15 obsahuje seznam, z nějž vymodelujeme rozevírací nabídku.

Výpis 5.15. Základ rozevírací nabídky

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
   <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
   <title>Události - Rozevírací nabídka</title>
   <link rel="stylesheet" href="styl.css" type="text/css" />

   <script src="js/jquery.js" type="text/javascript"></script>
   <script src="js/priklad_5.2.js" type="text/javascript"></script>
</head>
<body>
   <div id="hlavni">
      <div id="top">
         <div id="menu_box">
            <ul id="menu">
               <li class="headlink"><a href="#">Domů</a>
                  <ul>
                     <li><a href="#">O nás</a></li>
                     <li><a href="#">Historie</a></li>
                     <li><a href="#">Kontakty</a></li>
                  </ul>
               </li>
               <li class="headlink"><a href="#">Novinky</a>
                  <ul>
                     <li><a href="#">Sanita</a></li>
                     <li><a href="#">Kování</a></li>
                     <li><a href="#">Dlažby</a></li>
                     <li><a href="#">Tmely</a></li>
                  </ul>
               </li>
               <li class="headlink"><a href="#">Produkty</a>
                  <ul>
                     <li><a href="#">Dveře, okna</a></li>
                     <li><a href="#">Lepidla</a></li>
                     <li><a href="#">Izolace</a></li>
```

```
                    <li><a href="#">Obklady</a></li>
                    <li><a href="#">Cement</a></li>
                 </ul>
              </li>
              <li class="headlink"><a href="#">Galerie</a>
              </li>
           </ul>
        </div>
     </div>

     <div id="obsah">
        <h1>Obsah stránky</h1>
     </div>
  </div>
</body>
</html>
```

Přichystali jsme si jednoduchou nabídku, ale musíme upravit její vzhled. Zatím se jedná o obyčejný seznam v jazyce HTML. Budeme potřebovat pravidlo stylu pro element s identifikátorem `menu_box`. Tento element ohraničuje celou naši nabídku a my jej odsadíme o 15 pixelů. Dále definujeme styly tak, abychom z našeho seznamu udělali vodorovnou nabídku. Pravidla těchto stylů definujeme pod nadřazeným elementem s identifikátorem `menu`. Našeho cíle dosáhneme snadněji, pokud nastavíme vlastnosti `list-style` hodnotu `none`, díky čemuž se nebudou v tomto seznamu zobrazovat odrážky. Nakonec nastavíme vzhled jednotlivých položek tak, aby naše nabídka vypadala úhledně. Přidáme tedy pravidla kaskádových stylů z výpisu 5.16.

Výpis 5.16. Stylování rozevírací nabídky

```
#menu_box{
   padding: 15px;
}

#menu, #menu ul {
   list-style: none;
}

#menu, #menu * {
   padding: 0; margin: 0;
}

#menu li.headlink {
   background-color: #1240AB;
   width: 100px;
   float: left;
   text-align: center;
}

#menu li.headlink a {
   color: #FFF;
   text-decoration: none;
   display: block;
```

```
    padding: 5px;
}

#menu li.headlink ul {
    background-color: #4177F3;
    width: 100px;
    display: none;
    position: absolute;
    text-align: left;
}
```

S pravidly kaskádových stylů působí naše nabídka na první pohled jako kompletní. Už v této chvíli můžeme klepat myší na jednotlivé položky v hlavním panelu naší nabídky. Když ale přesouváme ukazatel myši nad jednotlivými položkami nabídky, nezobrazují se nám další podnabídky. Abychom dosáhli tohoto efektu, přidáme krátký kód knihovny jQuery z výpisu 5.17.

Výpis 5.17. Ošetření událostí myši

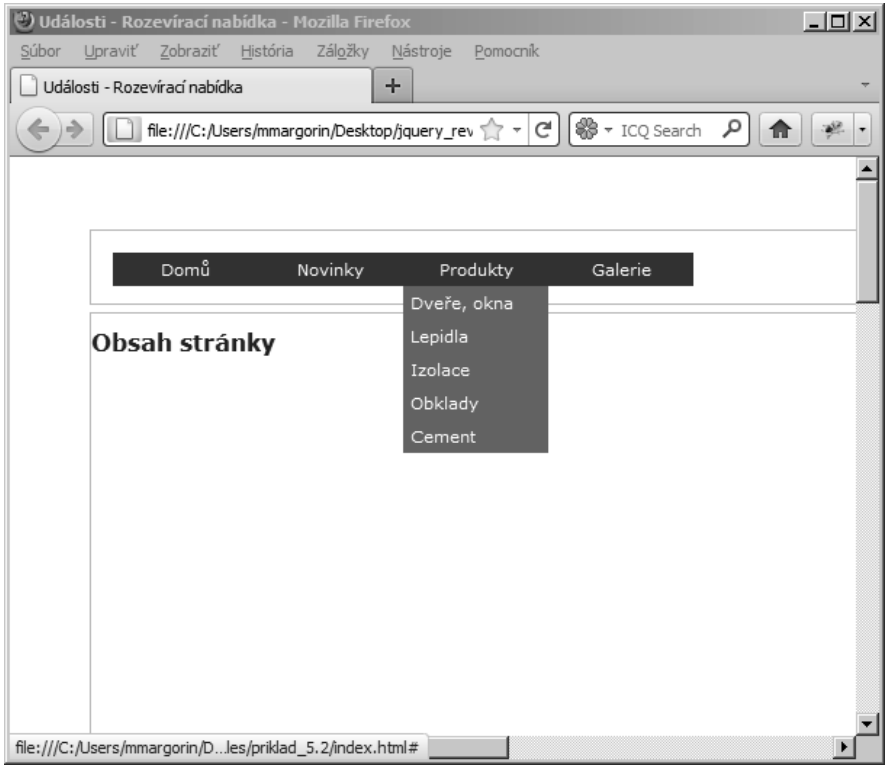
```
$(document).ready(function($) {
    $('li.headlink').hover(
        function() {$('ul', this).css('display', 'block');},
        function() {$('ul', this).css('display', 'none');}
    );
});
```

Jak jste se mohli přesvědčit, vytvoření jednoduché vyskakovací nabídky je s knihovnou jQuery opravdu snadné. Mnohem více práce bylo nutné věnovat na vytvoření seznamu v jazyce HTML a jeho stylování prostřednictvím jazyka CSS.

Samotný kód v knihovně jQuery je jednoduchý a krátký, a to zejména proto, že jsme zavolali metodu `hover()`. Tuto metodu jsme zavolali jen na elementy `li` s třídou `headlink`. Metoda `hover()` má dva parametry:

- Prvním parametrem je funkce pro obsluhu události `mouseenter` a druhým je funkce pro zpracování události `mouseleave`. V tomto případě jsme jako první argument předali funkci, v níž nastavujeme u všech podřízených elementů `ul` vlastnost `display` jazyka CSS na hodnotu `block`. Tím docílíme zobrazení skrytých položek seznamu pro nadřazený element `li.headlink`.
- Jako druhý argument předáváme funkci, ve které změníme hodnotu vlastnosti `display` pro stejné elementy na hodnotu `none`, čímž dříve zobrazené položky opět schováme. Jednoduše řečeno – při přesunu ukazatele myši nad element se zobrazí jeho dceřiné elementy `ul`. Jakmile tento element opustíme, zobrazené elementy `ul` se opět schovají.

Výslednou rozevírací nabídku můžete vidět na obrázku 5.3.



Obrázek 5.3. Rozevírací nabídka v knihovně jQuery

Jaké události myši máme k dispozici

Stejně jako jsme si popsali události prohlížeče, si nyní popíšeme události myši. Zároveň si ukážeme několik krátkých příkladů demonstrujících použití některých událostí.

- Událost `mousedown` – tato událost vzniká při stisknutí tlačítka myši.
- Událost `mouseup` – tato událost vzniká při uvolnění stisknutého tlačítka myši.
- Událost `mouseenter` – tato událost nastává ve chvíli, kdy ukazatel myši přesuneme na daný element z vnějšího elementu.
- Událost `mouseleave` – tato událost vzniká tehdy, když ukazatelem myši opustíme daný element z vnitřního elementu.

- Událost `mouseover` – tato událost nastává v okamžiku, kdy ukazatel myši přesuneme nad daný element.
- Událost `mouseout` – tato událost vzniká, když ukazatel myši opustí daný element.
- Událost `mousemove` – tato událost vzniká při pohybu ukazatele myši nad elementem.
- Událost `click` – tato událost vzniká při klepnutí myší na daný element.
- Událost `dblclick` – tato událost nastává při poklepání myší na daný element.

Příklad kombinace více událostí myši

V příkladu 5.2 jsme si ukázali, jak používat metodu `hover()`, která zpracovává události `mouseenter` a `mouseleave`. Může ale nastat situace, v níž není vhodné používat metodu `hover()`. Abychom si dokázali představit, jak tyto dvě události fungují samostatně, předvedeme si je na krátkém příkladu.

Příklad použití obou těchto událostí najdete ve výpise 5.19. Pro úplnost si připomeneme, že událost `mouseenter` vzniká pouze tehdy, když přesouváme ukazatel myši z vnějšího elementu. Naopak událost `mouseleave` nastává, jakmile opustíme ukazatelem myši daný element. V našem příkladu vytvoříme krabičky, u nichž obsloužíme uvedené události. Krabičky budeme simulovat pomocí bloků vytvořených z elementů `div`.

Tyto jednoduché krabičky vytvoříme ve standardním jazyce HTML tak, jak je patrné z výpisu 5.18.

Výpis 5.18. Dvě krabičky vytvořené v běžném jazyce HTML

```
<div id="vnejsi" style="width: 250px; height: 150px;
     background: #D6EDFC;"><p>Událost mouseenter</p>
   <div id="vnitrni" style="width: 200px; height: 100px;
     background: #FFCC00;"><p>Událost mouseleave</p></div>
</div>
```

Abychom zachovali jednoduchost, vložili jsme pravidla kaskádových stylů, s nimiž upravujeme vzhled obou elementů, přímo do zápisu těchto elementů. Po spuštění tohoto příkladu uvidíte, jak se mění text v obou krabičkách v závislosti na tom, jak posouváte ukazatel myši. Povšimněte si, že událost `mouseenter` nevzniká v případě, že přesouváte ukazatel myši z vnitřní krabičky označené oranžovou barvou do vnější krabičky. Stejně tak nevzniká událost `mouseleave`, když přesouváte ukazatel myši z vnější krabičky do vnitřní krabičky.

Výpis 5.19. Použití událostí mouseenter a mouseleave v knihovně jQuery

```
$(document).ready(function() {
   $("div").mouseenter(function() {
      $("p:first",this).text("Zachycena událost mouseenter");
   }).mouseleave(function(){
      $("p:first",this).text("Zachycena událost mouseleave");
   });
});
```

Na konci této velmi zajímavé části kapitoly, ve které jsme se věnovali událostem, si předvedeme zpracování událostí `click` a `dblclick`. Pouze si připomeneme, že událost `click` vzniká při klepnutí myší na daný element a událost `dblclick` při poklepání na daný element. Tyto události si ukážeme na prostém bloku textu. Blok obsahující náš testovací text vytvoříme v jazyce HTML jako na výpise 5.20.

Výpis 5.20. Blok obsahující testovací text vytvořený v jazyce HTML

```
<div style="width: 250px; background: #D6EDFC;">
<p>Uživatelé knihovny jQuery napsali spoustu užitečných návodů, díky nimž
se snadněji naučíte pracovat s touto knihovnou. Opět platí, že pokud
narazíte na špatně opsané vlastnosti knihovny jQuery, dejte nám prosím
vědět. Měli byste se možná porozhlédnout ve zbývající části dokumentace,
která je velmi komplexní a zahrnuje všechny aspekty knihovny jQuery. Pokud
máte jakékoliv dotazy, nebojte se zeptat v diskuzi.</p>
</div>
```

Opět jsme si řešení zjednodušili tak, že jsme vložili definice stylů přímo do daného elementu `div`. Po spuštění našeho příkladu můžeme pozorovat, jak se mění barva pozadí, a to podle toho, zda klepneme na tento blok jednou nebo dvakrát. Pokud klepneme jednou, vyvoláme událost `click` a změníme tím barvu pozadí na červenou. Jestliže klepneme dvakrát, vyvoláme událost `dblclick` a změníme barvu pozadí na modrou. Zpracování tohoto příkladu v knihovně jQuery se nachází ve výpise 5.21.

Výpis 5.21. Použití události click a dblclick v knihovně jQuery

```
$(document).ready(function() {
   $("div").dblclick(function () {
      $("p:first").css("background", "#0000FF");
   });

   $("p:first").click(function () {
      $(this).css("background", "#FF0000");
   });
});
```

Události klávesnice

Na závěr si probereme události klávesnice. Známe dva typy událostí klávesnice:

- Události, které vznikají, když stiskneme klávesu.
- Událost, která reaguje na psaní textu v textových polích.

Všechna písmena v textu mohou být jak malá, tak velká. Pokud chceme zjistit, zda uživatel stiskl malé nebo velké písmeno, můžeme použít událost `keydown`; případně událost `keyup`. Kdybychom potřebovali zjistit, jaký znak se zobrazil na obrazovce, použili bychom událost `keypress`. Vyzkoušíme si stručný příklad, jenž demonstruje použití událostí klávesnice. Tento příklad bude obsahovat textová pole a naší úlohou bude ošetřit vstup uživatele, jakmile začne vyplňovat tato pole. U takových formulářových elementů musíme tedy reagovat na jakékoliv stisknutí klávesy. Tyto elementy si přichystáme ve výpise 5.22.

Výpis 5.22. Příprava formuláře

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
   <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
   <title>Události klávesnice - Validace</title>
   <link rel="stylesheet" href="styl.css" type="text/css" />

   <script src="js/jquery.js" type="text/javascript"></script>
   <script src="js/priklad_5.3.js" type="text/javascript"></script>
</head>
<body>
   <h1>Události klávesnice - Validace</h1>
   <form action="#">
      <p>
         <label for="jmeno">Jméno</label>
         <input type="text" id="jmeno" name="jmeno" value="" />
         <span id="jmeno_log" style="color:red"></span>
      </p>
      <p>
         <label for="heslo">Heslo</label>
         <input type="password" id="heslo" name="heslo" value="" />
         <span id="heslo_log" style="color:red"></span>
      </p>
      <p>
         <label for="email">E-mail</label>
         <input type="text" id="email" name="email" value="" />
         <span id="email_log" style="color:red"></span>
      </p>
      <p>
         <input class="odeslat" type="submit" value="Odeslat" />
      </p>
```

```
    </form>
</body>
</html>
```

Kontrola uživatelského vstupu

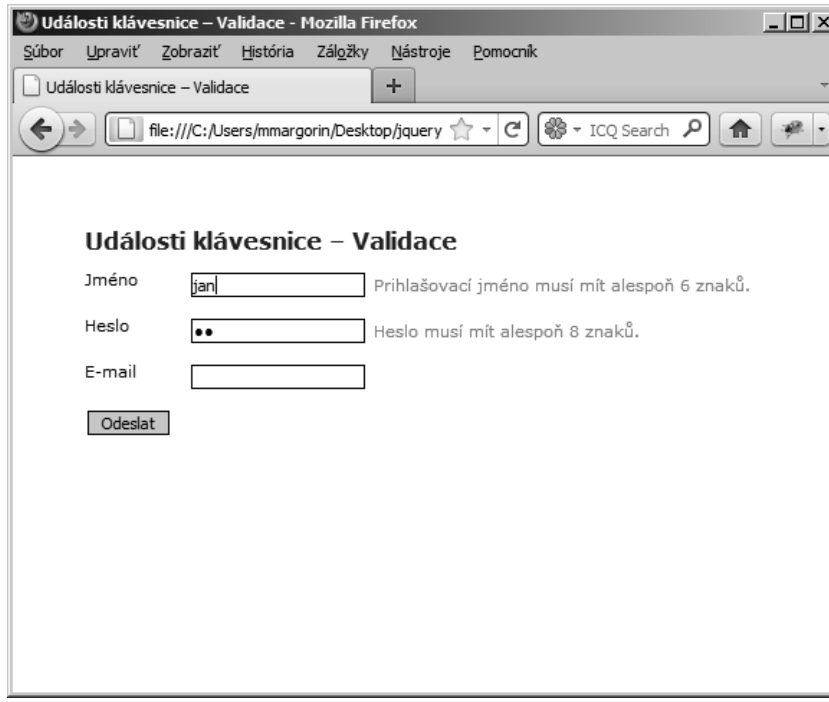
Připravili jsme si jednoduchý formulář a nyní napíšeme skript v knihovně jQuery, v němž budeme kontrolovat uživatelské vstupy. Vedle kontrolovaného pole se zobrazí chybová zpráva hned jak uživatel zadá nepovolenou hodnotu. Pro uživatele je takové chování formuláře velmi příjemné, protože přesně ví, kde udělal chybu. Kontrola formuláře se nachází ve výpise 5.23.

Výpis 5.23. Ošetření událostí klávesnice

```
$(document).ready(function($) {
    $("#jmeno").keypress(function() {
        $("#jmeno_log").empty();

        if ($("#jmeno").val().length < 6)
            $("#jmeno_log")
                .append("Prihlašovací jméno musí mít alespoň 6 znaků.");
    });

    $("#heslo").keypress(function() {
        $("#heslo_log").empty();

        if ($("#heslo").val().length < 8)
            $("#heslo_log").append("Heslo musí mít alespoň 8 znaků.");
    });

    $("#email").keypress(function(event) {
        $("#email_log").empty();

        if (event.which == '32')
            $("#email_log").append("Mezera není povolená.");
    });
});
```

Na kontrolu formuláře jsme použili událost `keypress`. Tuto událost zpracováváme pro každé textové pole v našem formuláři. Nejprve ošetřujeme textové pole s identifikátorem `jmeno`. V reakci na událost `keypress` nejdříve vymažeme obsah elementu s identifikátorem `jmeno_log` a následně spočítáme, kolik znaků uživatel zadal. Pokud zadal méně než šest znaků, vložíme do elementu s identifikátorem `jmeno_log` chybovou zprávu. Posléze ošetříme podobně textové pole s identifikátorem `heslo`. Nakonec se postaráme o poslední pole, do nějž uživatel vkládá e-mailovou adresu. V tomto případě nechceme, aby uživatel vložil mezeru, proto budeme kontrolovat každé stisknutí klávesy. Znak mezery má v programovacím jazyku JavaScript číslo 32. V našem

příkladu jsme tedy použili podmínku, v níž ověřujeme, zda uživatel stiskl klávesu s kódem 32. V kladném případě přidáme do elementu s identifikátorem `email_log` chybovou zprávu.



Obrázek 5.4. Validace pomocí událostí klávesnice v knihovně jQuery

Další metody klávesnice

Knihovna jQuery poskytuje také jiné metody, s nimiž je možné zpracovávat události klávesnice. Nejdůležitější z nich si uvedeme zde:

- Metoda `focusin()` – událost `focusin` vzniká tehdy, když zaměříme konkrétní element.
- Metoda `focusout()` – událost `focusout` nastává, jakmile daný element ztratí zaměření.
- Metoda `keydown()` – událost `keydown` vzniká, když uživatel stiskne klávesu na klávesnici.
- Metoda `keyup()` – událost `keyup` nastává, jakmile uživatel uvolní klávesu na klávesnici.