Steve Suehring

JavaScript

krok za krokem

- Od úplných základů po první webové aplikace
- Moderní výklad, přesné postupy, užitečné příklady
- Interaktivní prvky stránky a kontrola formulářů
- Překreslení obrázků, změna CSS, základy AJAXu

Ke stažení zdrojové kódy a soubory všech příkladů i cvičení

CPRESS Microsoft

Steve Suehring

JavaScript
Krok za krokem

Computer Press, a.s.
Brno
2008

JavaScript
Krok za krokem

Steve Suehring

Computer Press, a.s., 2008. Vydání první.

Překlad: Jakub Zemánek
Jazyková korektura: Pavel Bubla
Vnitřní úprava: Martina Petrová
Sazba: Martina Petrová
Rejstřík: René Kašík
Obálka: Martin Sodomka

Komentář na zadní straně obálky: Martin Domes
Technická spolupráce: Jiří Matoušek, Zuzana Šindlerová, Dagmar Hajdajová
Odpovědný redaktor: Martin Domes
Technický redaktor: Jiří Matoušek
Produkce: Daniela Nečasová

Computer Press, a. s.,
Holandská 8, 639 00 Brno

Objednávky knih:
http://knihy.cpress.cz
distribuce@cpress.cz
tel.: 800 555 513

ISBN 978-80-251-2241-9
Prodejní kód: K1615
Vydalo nakladatelství Computer Press, a. s., jako svou 3035. publikaci.

Stručný obsah

Část I

JavaCo? Vše ohledně JavaScriptu **17**

Část II

Použití JavaScriptu v praxi **91**

Část III

Integrace JavaScriptu do uživatelského rozhraní **173**

Část IV

Obsah

ČÁST II

POUŽITÍ JAVASCRIPTU V PRAXI

Kapitola 6

Kapitola 7

Kapitola 8

Kapitola 9

Kapitola 10

Část III

Integrace JavaScriptu do uživatelského rozhraní

Kapitola 19

Příloha

Úvodem

JavaScript je základním jazykem vývoje webových aplikací. Ať už svým webovým stránkám přidáváte interaktivitu nebo vytvářítе celé aplikace, dnešní web by nebyl bez JavaScriptu tím, čím je.

JavaScript je na standardech založený jazyk s formální specifikací. Nicméně, jak by vám řekl jakýkoli webový vývojář, každý z dnešních webových prohlížečů interpretuje tuto specifikaci trochu jinak. To samozřejmě práci webovým vývojářům ztěžuje. Naštěstí většina webových prohlížečů má v podpoře a interpretaci základních funkcí JavaScriptu sbíhavou tendenci.

Tato kniha poskytuje úvod do JavaScriptu, včetně popisu základních funkcí, stejně jako nejnovějších možností a přístupů, jako je Asynchronous JavaScript and XML (AJAX). Dnešní uživatelé webu se spoléhají na mnoho různých platforem a mnoho různých prohlížečů, takže v knize uvidíme snímky obrazovky z více různých prohlížečů a důraz bude kladen spíše než vlastní použití JavaScriptu směrem k použití založenému na standardech.

V první části knihy se s JavaScriptem seznámíte. Tato část vám pomůže začít s vývojem v JavaScriptu. K vývoji v JavaScriptu přitom nebudete potřebovat žádné speciální nástroje, ale uvidíte, jak lze k vývoji využít vývojová prostředí Microsoft Visual Studio, Eclipse, nebo prostě editor Poznámkový blok (či jakýkoli jiný editor). Dále prozkoumáte základy jazyka a funkce JavaScriptu. Poté zjistíte, jaký je vztah mezi JavaScriptem a webovými prohlížeči. A nakonec se vám v knize v plné kráse předvede AJAX. Naučíte se jej používat k vytvoření dynamických vyhledávacích formulářů.

Jak je kniha napsána

V této knize se krok za krokem naučíte programovat v JavaScriptu. Začnete od úplných základů a své vědomosti a znalosti programování v JavaScriptu budete zdokonalovat v každé kapitole pomocí praktických příkladů a cvičení.

Pokud už znáte základy JavaScriptu, potom budete moci první část knihy přeskočit. Kapitola 1, *JavaScript umí mnohem více, než si myslíte*, se podrobně věnuje určitému historickému pozadí, stejně tak jako základním předpokladům pro studium s touto knihou; obojí je přitom podkladem pro výuku v dalších kapitolách knihy. Kapitola 2, *Programujeme v JavaScriptu*, ukazuje, jak začít s JavaScriptem programovat. Pokud už znáte základy webového vývoje, můžete tuto kapitolu přeskočit. Každopádně se ale v obou kapitolách dozvíte vše podstatné o základech vývoje programů v JavaScriptu.

Kniha obsahuje obsah, jenž vám rychle pomůže určit polohu části, kterou hledáte. Každá kapitola obsahuje detailní seznam toho, co obsahuje.

Ke cvičením v knize si můžete navíc stáhnout startovací zdrojové kódy. Tento balík kódů navíc obsahuje i další kódy probírané v knize, takže příklady snadno otestujete, aniž byste museli kód opisovat.

Konvence použité v knize

Konvence	Co znamená
Číslované postupy	Cvičení krok za krokem s každým krokem pečlivě očíslovaným a vysvětleným. Každý postup začíná krokem 1.
Tip/Poznámka	Tyto speciální odstavce poskytují další informace, díky nimž si rozšíříte povědomí o probíraném tématu.

Konvence	Co znamená
Rozšiřující informace	Orámované části se zaměřují na další zdroje informací zvláště z programátorské praxe, které vám mohou pomoci ve vlastním vzdělávání.
Názvy souborů a složek	Názvy souborů a složek jsou odlišeny kurzivou.
`Kód JavaScriptu / zvýrazněný kód`	Jak kód v textu, tak celé výpisy zdrojového kódů JavaScriptu jsou formátovány neproporcionálním písmem. Probíraný kód je zvýrazněný navíc tučností.
Uživatelský vstup	Texty, které má čtenář zapsat, jsou zvýrazněny tučností.

Zdrojové kódy cvičení a příkladů

Zdrojové kódy, jak startovací, tak finální kódy ke všem cvičením, ale také kódy příkladů si čtenáři mohou stáhnout z webové stránky knihy na adrese *http://knihy.cpress.cz/k1615*. Všechny kódy byly pečlivě lokalizovány do češtiny, takže je snáze pochopí také úplní začátečníci.

Balík kódů si stáhněte a rozbalte pomocí některého z archivačních programů, například WinRAR nebo WinZIP. Protože JavaScript je obvykle závislý na okolní webové stránce, zdrojové kódy pro cvičení krok za krokem byly rozděleny do složek podle kapitol. To vám umožní zkopírovat a vložit opakující se kód HTML do svých dokumentů, čímž se budete moci v rámci cvičení soustředit pouze na JavaScript.

Každá složka s kódy kapitoly taktéž obsahuje složku *DokonceneKody*, v níž najdete kompletní příklady. Tyto soubory můžete otevřít tak, jak jsou a prozkoumat tak okamžitě příklady ze všech kapitol.

Poznámka redakce českého vydání

Nakladatelství Computer Press, které pro vás tuto knihu přeložilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

Computer Press
redakce PC literatury
Holandská 8
639 00 Brno

nebo

knihy@cpress.cz

Další informace a případné opravy českého vydání knihy najdete na internetové adrese *http://knihy.cpress.cz/k1615*. Prostřednictvím uvedené adresy můžete též naší redakci zaslat komentář nebo dotaz týkající se knihy. Na vaše reakce se srdečně těšíme.

Část I

JavaCo?
Vše ohledně JavaScriptu

V této části:

Kapitola 1

JavaScript umí mnohem více, než si myslíte

Vítejte v této úvodní kapitole. Po jejím přečtení budete schopni:

- Vyznat se v minulosti JavaScriptu
- Rozeznat část programu napsanou v JavaScriptu
- Používat javascriptový pseudoprotokol
- Umístit kód JavaScriptu do webové stránky tam, kam patří
- Porozumět tomu, co JavaScript umí a co ne

Trocha z historie JavaScriptu

JavaScript není Java. S povědomím o této skutečnosti se můžete přesunout k větším a důležitějším věcem, například jak vytvořit bezvadnou vysouvací navigační nabídku. Ale vážně, JavaScript je hovorový název pro specifikaci formálně nazývanou jako ECMAScript. Ale název ECMAScript není nic úchvatného. Jen to zkuste, řekněte nahlas: „ECMAScript". Vidíte? (O ECMAScriptu si povíme později.)

Odkud vlastně JavaScript vzešel? Pravděpodobně neznáte velmi bohatou a zajímavou historii JavaScriptu – a popravdě, nejspíš o ni ani tak moc nestojíte. Pokud je tohle váš případ, možná máte pokušení přeskočit dopředu na další kapitolu a vrhnout se rovnou mezi kódy JavaScriptu. To by ale byla samozřejmě chyba, neboť byste přišli o ten parádní text věnující se JavaScriptu dále. A mimo to, pochopení trochy minulosti JavaScriptu je důležité k pochopení toho, jak je tento jazyk dnes implementován do různých prostředí.

JavaScript byl původně vyvinut Brendanem Eichem ve společnosti NetScape mezi léty 1995 až 1996. V té době se tento jazyk nazýval LiveScript. To ale nebylo skvělé jméno pro zcela nový jazyk, a tak zde příběh nekončí. Jedním nešťastným rozhodnutím si lidé z marketingu postavili hlavu a jazyk pojmenovali JavaScript. Zmatení se dostavilo brzy. Jistě víte, že jazyk Java byl v té době horkou novinkou a někdo se zkrátka rozhodl parazitovat na jeho popularitě. Ve výsledku se JavaScript sám o sobě usadil vedle jazyka Java. Naneštěstí jazyk Java, jehož popularita vycházela z častého používání, si vysloužil špatnou pověst, neboť některé z webových stránek používaly Javu k prezentaci nebo k přidání zbytečných vylepšení (jako byl například pohybující se text). Zkušenost uživatelů utrpěla, protože Java vyžadovala načtení zásuvného modulu (plug-inu) ve webovém prohlížeči, čímž zpomalovala proces surfování a celkově návštěvníkům stránek způsobovala potíže, a to se nezmiňujeme o problémech s přístupností. JavaScript byl neúprosně spojován s Javou a s jejími problémy. Až později toto špatné propojení obou jazyků JavaScript setřásl.

JavaScript není kompilovaným (překládaným) jazykem, čímž vytváří pocit, že není dostatečně výkonným jazykem. Mnozí weboví programátoři v závěru devadesátých let minulého století

nejprve JavaScript odmítli, ale brzy na to objevili jeho užitečnost a sílu, díky níž lze jak simulovat, tak vytvářet interaktivitu na Word Wide Webu. Do té doby se mnoho webových stránek vytvářelo pomocí hypertextového značkovacího jazyka (HTML) s grafikou, jež postrádala jak vizuální dojem, tak schopnost ovlivňovat obsah stránek.

JavaScript se brzy zaměřil na ověřování obsahu formulářů na straně klienta a práci s obrázky na webových stránkách, aby poskytl základní, ale užitečnou, interaktivitu a zpětnou vazbu návštěvníkům. Když uživatel webových stránek vyplnil formulář, JavaScript se mohl použít k okamžité kontrole obsahu webového formuláře, aby se tak data nemusela odesílat webovému serveru. To byl skvělý způsob, jak zrychlit aplikace a nechat je reagovat na podněty uživatele, a tím uspořit čas běžně nutný ke komunikaci se serverem. A to zvláště v době, kdy se ještě nerozšířil širokopásmový Internet (broadband).

Na scénu vstupuje Internet Explorer 3

Do třetí verze Microsoft Internet Exploreru vydané v roce 1996 Microsoft zahrnul podporu jádra JavaScriptu, známou jako JScript, společně s podporou dalšího skriptovacího jazyka, VBScriptu. Naneštěstí, ačkoliv tyto jazyky byly stejné, nebyly zcela stejné ve své implementaci. Proto se vyvinuly metody detekující, který prohlížeč používá návštěvník webových stránek, podle čehož mu poskytl takový JavaScript, jenž v daném prohlížeči fungoval. Tento proces je známý jako detekce prohlížeče a budeme se o něm bavit v kapitole 14 nazvané *Prohlížeče a JavaScript*. Detekce prohlížeče se stále používá, i když stále v menší míře.

A pak se objevil ECMAScript

V polovině roku 1997 společnosti Microsoft a Netscape spolupracovaly v rámci skupiny European Computer Manufactures Association (ECMA) na první verzi ECMAScriptu. Od té doby všechny prohlížeče od společností Netscape a Microsoft implementují jednotlivé verze standardu ECMAScript. Jiné populární prohlížeče, jako Firefox a Opera, taktéž implementují standard ECMAScript.

Poslední verze ECMAScriptu přijatá jako standard známý pod jménem ECMA-262 byla vydána v roce 1999. Dobrou zprávou je, že prohlížeče jako je Internet Explorer 4 a Netscape 4.5 tento standard podporovaly a každý důležitý prohlížeč od té doby podporuje verzi JavaScriptu přijatou ve standardu ECMA-262. Špatnou zprávou je, že každý z prohlížečů si tento standard vykládá trochu jiným způsobem. To znamená, že i dnes se vývojáři v JavaScriptu stále potýkají s nekompatibilitou.

Na čtvrtém vydání ECMAScriptu se stále pracuje, ale brzy bude vydán. Podívejte se na http://www.ecma-international.org – zde získáte o standardu více informací.

Jako vývojáři, kteří budete JavaScript včleňovat do svých webových aplikací, budete muset počítat s rozdíly v interpretaci JavaScriptu. To znamená, že budete muset implementovat skripty trochu jiným způsobem, a také to znamená, že je budete muset testovat, testovat a testovat znovu v různých prohlížečích a na různých platformách. Na dnešním Internetu existuje jen velmi malá tolerance špatně navržených aplikací, jež pracují jen v určitém prohlížeči.

Důležité: Je velmi důležité testovat vaše webové stránky ve více prohlížečích, a to i když si myslíte, že vaši webovou aplikaci nebudou používat lidé, kteří nevyužívají Internet Explorer. I když si budete jistí, že se vaše aplikace bude používat výhradně v Internet Exploreru, nebo je to jediný prohlížeč, jejž oficiálně podporujete, stále byste ji měli testovat v jiných prohlížečích. Nejen z důvodu bezpečnosti, ale také proto, že se ukáže, zda jste vývojáři, kteří rozumí dnešním internetovým technologiím.

Tolik standardů ...

Pokud byste se domnívali, že standardy okolo programování v JavaScriptu jsou poměrně slabě definované, měli byste pravdu. Každý prohlížeč podporuje JavaScript trochu jinak, čímž budou vaši práci ztěžovat. Pokoušet se popsat všechny tyto rozdíly je tedy obtížnější, než kdyby byl tento jazyk jednoduchou, specifickou, entitou, jako jsou třeba jednotlivé verze Microsoft Visual Basicu či Perlu. Ale není, a tak vaší (i mou) prací bude na všechny tyto rozdíly myslet, nezbytně s nimi počítat a pokusit se najít nejmenší společný jmenovatel jak jen to bude možné.

DOM

Jiným standardem, který používají programátoři v JavaScriptu, je objektový model *dokumentu* (DOM – Document Object Model) vyvinutý konsorciem World Wide Web Consortium (W3C). W3C popisuje DOM jako „na platformě – a jazyku – nezávislé rozhraní umožňující programům a skriptům dynamicky přistupovat a aktualizovat obsah, strukturu a styly dokumentů". Pro vás to jednoduše znamená, že existuje prohlížeči používaná specifikace, kterou můžete použít k práci s webovou stránkou v dynamickém slova smyslu. DOM reprezentuje dokumenty HTML a XML (Extensible Markup Language – rozšiřitelný značkovací jazyk) ve formě stromové struktury a umožňuje dynamický přístup k objektům této struktury. JavaScript silně souvisí s modelem DOM a nabízí pro práci s ním mnoho funkcí.

Podobně jako JavaScript, i DOM interpretovaly různé prohlížeče rozdílně, čímž dále ztížily život programátorům v JavaScriptu. Internet Explorer 4 a předchozí verze prohlížeče NetScape v sobě zahrnovaly podporu pro prvotní model DOM známý jako DOM úrovně 0. Jestliže použijete DOM této úrovně, můžete si být celkem jisti podporou u všech těchto prohlížečů a jejich následovníků.

Internet Explorer ve verzích 5 a 5.5 zahrnuly částečnou podporu DOM úrovně 1, zatímco Internet Explorer 6 a pozdější verze podporují již úroveň 2. Ostatní prohlížeče, jako Firefox a Opera, plně podporují standardy W3C.

Jestliže je zde něco, čemu byste se při výuce standardů JavaScriptu a příbuzných standardů DOM neměli vyhnout, pak to, že musíte věnovat pozornost programovému kódu, který vytváříte (asi žádné překvapení), a syntaxi tohoto programového kódu. Pokud totiž nebudete, může programový kód snadno selhat a způsobit, že se vaše stránka nevykreslí v daném prohlížeči. Kapitola 10, *Objektový model dokumentu*, pokrývá model DOM mnohem více do detailu.

Tip: W3C poskytuje aplikaci, jež umožňuje testování vašeho webového prohlížeče pro podporu různých úrovní modelu DOM. Tuto aplikaci naleznete na adrese http://www.w3.org/2003/02/06-dom-support.html.

Co obsahuje program v JavaScriptu

Program napsaný v JavaScriptu se skládá z příkazů tvořených znaky, operátory a identifikátorů umístěných za sebou v řadě. Toto uspořádání pak dává smysl interpreteru JavaScriptu, který obsahuje většina webových prohlížečů. Vypadá to složitě, ale ve skutečnosti na tom není nic komplikovaného pro kohokoliv, kdo programoval v jakémkoli jiném jazyku. Příkaz může vypadat například takto:

```
var maleCislo = 4;
```

Tento příkaz obsahuje znaky, v tomto případě rezervované slovo `var`, následované dalšími znaky, jako je identifikátor, operátor a číslo. O těchto částech příkazu se naučíte více v kapitole 2, *Programujeme v JavaScriptu*, a dále v knize. Účelem tohoto příkazu je nastavit proměnnou `maleCislo` na hodnotu 4.

Stejně jako v jakémkoli jiném programovacím jazyce příkazy uspořádáváte za sebe způsobí, že program vykoná jednu nebo více funkcí. Mluvíme-li o funkcích, je třeba říct, že JavaScript má svůj vlastní způsob, jímž funkce definuje a o němž se dočtete více v kapitole 7, Práce *s funkcemi*. JavaScript nabízí řadu vestavěných funkcí, jež můžete použít ve vašich programech.

Použití pseudoprotokolu a funkce javascript

1. Otevřete webový prohlížeč, například Internet Explorer nebo Firefox.
2. V adresním řádku napište následující kód a stiskněte klávesu Enter:

```
javascript:alert("Ahoj světe");
```

3. Po stisku klávesy Enter uvidíte dialog podobný tomuto:

Gratuluji! Právě jste naprogramovali váš první (ačkoliv nepříliš užitečný) kousek programového kódu v JavaScriptu. Nicméně na tomto kousku kódu můžete vidět dvě důležité věci, které využijete ve své snaze programovat v JavaScriptu: identifikátor pseudoprotokolu `javascript`, a velmi užitečnou funkci `alert`. Každé z těchto dvou položek se budeme věnovat v dalších kapitolách, ale nyní je dostačující to, že jste se naučili něco, co využijete v budoucnosti!

JavaScript je taktéž řízen událostmi, čímž mám na mysli to, že může reagovat na určité události nebo „věci, které se dějí", jako je klepnutí myší či změna textu ve formulářovém poli. Propojení JavaScriptu s událostí je pro mnoho způsobů a běžné užití JavaScriptu zcela zásadní. V kapitole 14 se dozvíte, jak použitím JavaScriptu reagovat na události.

Vložení kódu v JavaScriptu do webové stránky

Pokud jste nováčky v HTML, vše, co nyní potřebujete znát, je to, že se v HTML vkládají do webové stránky elementy pomocí párů značek uzavřených do ostrých závorek. A že uzavírací značka začíná lomítkem (/). Samotné elementy je pak možné vnořovat do sebe. Programový kód v JavaScriptu se vkládá do těla značky `<script>`, jež se umisťuje dovnitř značky `<head>`, a ta zase dovnitř značky `<body>`. Zde je příklad:

```
<html>
  <head>
    <title>Titulek webové stránky</title>
    <script type="text/javascript">
        // Sem patří programový kód v JavaScriptu
    </script>
  </head>
  <body>
    <script type="text/javascript">
    // Sem programový kód v JavaScriptu patří také
    </script>
  </body>
</html>
```

Programový kód v JavaScriptu umístěný do značky <body> se spouští, jakmile na něj při čtení dokumentu dojde řada. To se může hodit, pokud potřebujete psát do dokumentu za pomoci nějaké funkce JavaScriptu, jak je vidět zde (volané funkce jsou zvýrazněné tučně):

```
<html>
  <head>
    <title>Titulek webové stránky</title>
    <script type="text/javascript">
```

```
      // Sem patří programový kód v JavaScriptu
    </script>
  </head>
  <body>
    <script type="text/javascript">
      document.write("Ahoj");
      document.write(" Ahoj");
    </script>
  </body>
</html>
```

Když používáte programový kód v JavaScriptu ve stránce XHTML (Extensible Hypertext Markup Language – rozšířený hypertextový jazyk), znaky < a & se interpretují jako řídicí znaky jazyka XML, což v JavaScriptu působí problémy. Tento problém lze vyřešit použitím následující syntaxe ve stránce XHTML:

```
<script type="text/javascript">
  <![CDATA[
  // Sem patří programový kód v JavaScriptu
  ]]>
</script>
```

Jasně, vypadá to škaredě. Nicméně, i pro tento zápis existuje řešení: použití externího souboru s programovým kódem v JavaScriptu. V kapitole dvě poznáte, jak se s tímto úkolem vypořádat.

Typy dokumentů

Pokud jste někdy alespoň trochu programovali pro web, pravděpodobně znáte deklarace typu dokumentu, neboli deklarace DOCTYPE, jak se také někdy nazývají. Jednou z důležitých věcí při návrhu webové stránky je zajistit, že bude na začátku stránky obsahovat přesnou a syntakticky správnou deklaraci DOCTYPE. Deklarace DOCTYPE, pro niž se běžně užívá zkratka DTD, umožňuje prohlížeči (nebo parsujícímu programu) poznat pravidla, jimiž se bude řídit při parsování (syntaktickém zpracování) elementů dokumentu.

Například deklarace DOCTYPE pro HTML 4.01 vypadá takto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
```

Pokud k vytváření webových projektů používáte Visual Studio 2005 nebo novější, každé stránce se přidělí automaticky deklarace DOCTYPE pro standard XHTML 1.0, takto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Pokud deklarace DOCTYPE vynecháte, prohlížeč bude stránku vyhodnocovat za použití nestandardního režimu známého jako Quirks Mode. Přepnutí prohlížeče do tohoto módu bude znamenat, že dokument bude nakonec vypadat jinak, než jste zamýšleli, zvláště v různých prohlížečích.

Pokud DOCTYPE deklarujete, je důležité se ujistit, že výsledné programové kódy dokumentů HTML, kaskádových stylů (Cascading Style Sheets – CSS) a JavaScriptu také odpovídají webovým standardům. To je důležité, pokud chcete zajistit, že se dokument bude zobrazovat tak, jak jste zamýšleli, různému publiku, bez ohledu na rozhraní nebo prohlížeč, který k prohlížení dokumentů toto publikum používá. Více se budeme validaci (kontrole) kódu HTML a CSS věnovat v této knize v kapitole 15, *JavaScript a CSS*. W3C poskytuje zdarma online validátor na adrese *http://validator.w3.org/*. Ten můžete použít ke kontrole jakékoli veřejně dostupné webové stránky.

Tip: Validátor kódu používejte pravidelně, dokud nebude dokument vzhledem ke standardům bezchybný, a vždy validujte jakýkoli dokument předtím, než webový projekt uveřejníte na Internetu.

Co JavaScript umí

JavaScript je velkou měrou komplementární jazyk, čímž myslím, že není běžné, aby byla celá aplikace napsána pouze v JavaScriptu bez pomoci jiných jazyků, jako třeba HTML, a bez prezentace ve webovém prohlížeči. Podporu JavaScriptu nabízí některé produkty společnosti Adobe, ale i tak se JavaScript primárně používá ve webovém programování.

Jak uvidíte (nebo už možná víte), JavaScript je také jako písmeno J součástí zkratky AJAX (Asynchronous JavaScript and XML – asynchronní JavaScript a XML), miláčka fenoménu Web 2.0. Mimo to je JavaScript každodenním jazykem poskytujícím očekávanou, nebo možná vyžadovanou, interaktivitu současnými webovými uživateli.

JavaScript umí na klientské straně aplikace spoustu věcí. Dokáže webovým stránkám přidat potřebnou interaktivitu – například vytvořením vysouvacích navigačních nabídek, transformací textu na webové stránce, dynamickým přidáním elementů do stránky a dokáže též pomoci se vstupními formulářovými poli.

Zbytek knihy se zabývá tím, co JavaScript dokáže. Ale nejprve se podívejme na několik věcí, které JavaScript prostě nedokáže. Všimněte si, že tento seznam nepostihuje úplně vše.

Co JavaScript neumí

Mnoho z toho, co JavaScript nedokáže, je výsledkem toho, že použití JavaScriptu je limitované prostředím webových prohlížečů. Tato část popisuje některé úkoly, jež JavaScript splnit nedokáže, a ty, které by umět provést ani neměl.

JavaScript nemůžete klientovi vnutit

V praxi to znamená, že JavaScript svou funkcionalitu opírá o jiné rozhraní nebo hostitelský program. Tímto hostitelským programem je běžně webový prohlížeč na straně klienta, taktéž známý jako uživatelský agent (user agent). Protože JavaScript je jazyk pracující na straně klienta, dokáže dělat jen to, co mu dovolí klient.

Někteří lidé stále používají starší prohlížeče, které zcela JavaScript nepodporují. A stále i další prohlížeče nebudou kvůli přístupnosti, čtečkám textu a jinému podpůrnému softwaru v prohlížeči schopné podporovat mnoho efektních možností, jež JavaScript nabízí. A někteří lidé zkrátka použití JavaScriptu v prohlížeči zakážou, už jen proto, že mohou, nebo protože se obávají problémů se zabezpečením (ať jsou reálné či nikoliv), či kvůli slabé reputaci JavaScriptu mezi některými lidmi, neboť je například obtěžují vyskakovací (automaticky otevíraná) okna s reklamou.

Bez ohledu na důvod potřebujete vyvinout úsilí navíc, abyste zajistili, že vámi navrhované webové stránky budou přístupné také těm jednotlivcům, kteří JavaScript nepoužívají. Nyní slyším vaše protesty: „Ale tahle možnost je přece [sem vložte váš vlastní superlativ: skvělá, příjemná, podstatná, krásná, nádherná]." Bez ohledu na to, jak krásná tato možnost může být, je šance, že byste mohli mít prospěch z lepší interoperability a zvýšené návštěvnosti stránek. V části *Tipy pro používání JavaScriptu,* později v této kapitole, vám nabídnu některé rady, jež můžete následovat, díky nimž zajistíte, že se JavaScript bude na vašich webových stránkách používat správně.

Možná pomůže nad celou záležitostí uvažovat odlišně. Když vytváříte webovou aplikaci fungující na webovém serveru IIS 6.0 (Internet Information Service), můžete si být celkem logicky jistí, že

tato aplikace by měla fungovat kdekoliv, kde je na serveru nainstalována služba IIS 6.0. Podrobně když vytvářítе aplikaci pro webový server Apache 2, můžete si být jistí, že bude fungovat na jiných instalacích Apache 2. To samé nicméně nemůžeme říct o JavaScriptu. Když píšete aplikaci, jež pracuje na vašem stolním počítači, nutně už nemusí fungovat jinému člověku na jeho počítači. Zkrátka nemůžete kontrolovat, jak vaše aplikace bude fungovat, jakmile se zašle klientovi.

JavaScript negarantuje bezpečnost dat

Protože programový kód v JavaScriptu běží celý na straně klienta, vývojář se musí naučit jak jej používat. Jak asi tušíte, má provádění kódu na straně klienta vážné bezpečnostní důsledky. Jakmile je jednou program v počítači klienta, klient může provádět dost nepěkné věci programovému kódu samotnému, než odešle výsledek zpět serveru. Tak jako v případě dalšího webového programování byste nikdy neměli věřit jakýmkoli datům přicházejícím od klienta. I když jste použili funkce JavaScriptu pro kontrolu obsahu formuláře, je pořád nutné, abyste provedli kontrolu těchto vstupů znovu ve chvíli, kdy se data dostanou na server. Klient s vypnutou podporou JavaScriptu může zpět poslat z webového formuláře nesmyslná data. Pokud uvěříte, s dostatečnou dávkou nevinnosti, že vaše javascriptové funkce už otestovaly data z formuláře k zajištění jejich správnosti, možná zjistíte, že tato chybná data vrácená serveru mají neočekávané, a možná dokonce nebezpečné, následky.

Důležité: Pamatujte na to, že podporu JavaScriptu může návštěvník stránek ve svém počítači vypnout. Roztomilý trik, jímž pomocí JavaScriptu znepřístupníte klepnutí pravým tlačítkem myši do stránky nebo návštěvníkům zabráníte zobrazovat zdrojový kód stránky, nemůže být spolehlivě funkční a neměli byste jej tedy používat jako bezpečnostní opatření.

JavaScript nemůže překročit hranice domény

Vývojář v JavaScriptu si musí být vědom tzv. *politiky stejného původu* (Same-Origin Policy), jež přikazuje, že skripty spuštěné v jedné doméně nesmí mít přístup k prostředkům jiné domény, ani nemohou ovlivňovat skripty a data v jiné doméně. Například JavaScript se může použít k otevření nového okna prohlížeče, ale obsah tohoto nového okna je vůči volajícímu skriptu poněkud omezen. Pokud stránka mého webu, braingia.org, obsahuje programový kód v JavaScriptu, nemůže tento přistupovat k prostředkům z jiné domény, jako je třeba microsoft.com. To je podstata politiky stejného původu: aby se mohl prostředek použít ve stránce, musí se programový kód v JavaScriptu spustit ze stejného umístění nebo z něj pocházet.

Politika stejného původu je často problematická při použití rámců nebo objektu XMLHttpRequest používaného v AJAXu, kdy je třeba pomocí JavaScriptu poslat více požadavků různým webovým serverům. Tohoto tématu se dotkneme v kapitole 18, *AJAX*. Nyní mějte na vědomí, že JavaScript je možné použít pouze v rámci okna prohlížeče.

JavaScript nepracuje na straně serveru

Když vyvíjíte kód na straně serveru, jako v případě jazyků Visual Basic.NET nebo PHP (PHP je rekurzivní zkratka, jež značí: „PHP: Hypertext Preprocessor"), celkem logicky si můžete být jistí, že server implementuje určité funkce, jako je komunikace s databází nebo poskytnutí přístupu k modulům nutným pro chod webové aplikace. JavaScript ovšem nemá k proměnným na straně serveru přístup. Například JavaScript nemůže přistupovat k databázi umístěné na serveru. Kód JavaScriptu je limitován tím, co se může udělat v rámci platformy, na níž skript běží, což je typicky prohlížeč.

Pokud jste obeznámeni s programováním na straně serveru, bude pro vás odlišné i to, že nikdy nebudete vědět, jaké funkce klient vůbec podporuje, aniž byste aplikaci testovali na mnoha klientech. Když programujete na straně serveru, a když server neimplementuje požadované

funkce, budete to zkrátka hned vědět, neboť skript na straně serveru selže, jakmile jej otestujete. Implementace podpory pro kód na straně serveru by se neměla bezdůvodně měnit, neboť pak lze snadněji poznat, jaký kód můžete psát a jaký ne. To samé ovšem nelze říct o kódu v JavaScriptu, jenž je zamýšlen tak, aby běžel na straně klienta, který je zcela mimo vaši kontrolu.

Tipy pro používání JavaScriptu

V rámci dobrého webového designu vstupuje do hry několik faktorů, ale popravdě, kdo rozhoduje o tom, co je dobré a co nikoliv? Jeden návštěvník může něco nazvat jako hnusný mix barev a textu jako byste je dali do kýble, zatřepali s ním a poté prostě vylili na stránku; jiný může váš design a barvy milovat.

Jelikož čtete tuto knihu, domnívám se, že hledáte nějakou pomoc s použitím JavaScriptu k vylepšení vašich webových stránek. Taktéž se domnívám, že chcete používat programovací jazyk, abyste lidem zpříjemnili a usnadnili používání vašich stránek a abyste vylepšili jejich vzhled a dojem, a aby zkrátka fungovaly lépe.

Design webu není a nikdy nebyl zcela objektivní záležitostí. Cíle webových stránek mohou být informační, čímž vyžadují volbu jednoho přístupu, zatímco cílem jiných webových stránek může být propojení s aplikací, a mohou tak vyžadovat specializovaný design a funkcionalitu. Nicméně, mnoho populárních a zdánlivě dobře navržených stránek obsahuje jisté společné aspekty. Pokusím se je zde shrnout, ačkoliv vás žádám, abyste si pamatovali, že jsem zde nevytvořil vyčerpávající seznam a že jde jednoduše o názor jednoho člověka.

Dobře navržené webové stránky dávají důraz na funkce před formou. Uživatel obvykle navštíví webové stránky, aby získal informace. Čím složitěji se na vašich stránkách dá orientovat, tím spíše se uživatel prostě přesune na jiné stránky s lepší navigací.

Animace a blikající části stránky přicházejí a odcházejí, ale co zůstává, jsou stránky poskytující základní informace prezentované profesionálním, jednoduše přístupným způsobem. Použití nejnovějšího animačního softwaru nebo webové technologie mně vždy připomene časy HTML značky `<blink>`. Značka `<blink>`, pro ty, kteří ji nikdy neviděli v akci, způsobuje, že do ní vložený text se na obrazovce objevuje a zase ztrácí. Téměř všichni weboví vývojáři, jak se zdá, nenávidí, co značka `<blink>` provádí. Ti stejní vývojáři by proto měli mít na mysli, že úchvatné animace nebo roztomilý efekt na webových stránkách bude zítra jen další značkou `<blink>`. Úspěšné webové stránky se drží základů a používají blikání pouze tehdy, když to obsah vyžaduje.

Používejte elementy, jako jsou mapa stránek, popisné značky a jednoduchou navigaci a nevyžadujte speciální software nebo zásuvné moduly (plug-iny) k zobrazení hlavního obsahu stránek. Příliš často navštívím webové stránky, na nichž se musím zastavit, protože potřebuji zásuvný modul či jeho poslední verzi nebo nějaký přehrávač (který nemám), abych se mohl na stránkách navigovat.

Ačkoliv mapy stránek, popisné značky a jednoduchá navigace mohou vypadat zvláštně, jde o nepostradatelné součásti přístupnosti. Čtečky textu a podobné technologie, které zpřístupňují stránky zrakově postiženým, tyto objekty používají a často mají s JavaScriptem velké problémy.

Dobře navržené webové stránky dodržují standardy. Webové standardy jsou zde proto, abychom je dodržovali, takže je můžete ignorovat pouze ke své vlastní škodě. Správné použití deklarace DOCTYPE a dobře strukturovaný kód HTML vám pomůže zajistit, že se stránky zobrazí návštěvníkům vašich stránek správně. Validace pomocí nástroje Markup Validator konsorcia W3C se velmi doporučuje. Pokud vaše stránka obsahuje chyby, opravte je!

Dobře navržené webové stránky se správně vykreslí v různých prohlížečích. I když měl kdysi Internet Explorer 90 % podílu na trhu, pro vývojáře nikdy nebylo dobré, když ignorovali jiné prohlížeče. Pokud tak činili, obvykle to znamenalo ignorování přístupnosti. Takže lidé s textový-

mi čtečkami či jinými doplňky nemohli stránky používat. Lidé používající jiný operační systém než Microsoft Windows měli při návštěvě stránek zkrátka smůlu.

I když je Internet Explorer mezi návštěvníky webových stránek stále lídrem, stále je velká možnost, že 2 až 3 z 10 uživatelů mohou používat jiný prohlížeč. Samozřejmě tato odchylka záleží především na objektu zájmu. Čím máte více technicky zaměřené publikum, tím více se budete muset přizpůsobit jiným prohlížečům než je Internet Explorer. Proto, když se vaše stránky věnují technickému publiku, budete potřebovat, aby vaše stránky fungovaly v prohlížečích Firefox, Safari nebo dokonce Lynx.

Bez ohledu na předmět zájmu webových stránek, nikdy byste se neměli odvrátit od uživatelů jen z důvodu jejich volby jiného prohlížeče. Představte si obchodníka, který se odvrátí od 3 z 10 potenciálních zákazníků jen kvůli jejich botám. Takový obchod by nemohl dlouho fungovat, nebo by alespoň neprosperoval.

Pokud se vrhnete do dodržování webových standardů, je šance, že budete dělat většinu toho, co potřebujete udělat k zajištění podpory více různých prohlížečů. Také platí, že pokud se vyhnete zásuvným modulům třetích stran nutných pro správný chod vašich webových stránek, zajistíte tím i jejich správné vykreslení.

Dobře navržené webové stránky používají odpovídající technologie v odpovídající čas. Pokud mluvíme o zásuvných modulech, platí, že dobře navržené webové stránky nepoužívají takových technologií ani příliš, ani málo. Webové stránky nabízející video nahrávky běžně vyžadují nějaký přehrávač k jejich přehrání. Podobně na stránkách nabízejících hudební obsah je běžné, že na pozadí přehrávají hudbu. Není to ale případ jiných stránek. Pokud cítíte, že vaše stránky potřebují hudbu na pozadí, vraťte se zpět k tabuli a prvně prověřte, proč potřebujete webové stránky. Chvěji se hrůzou při pomyšlení na právní stránky, jež jsem jednou navštívil. Stránky začínaly přehráváním firemní znělky na pozadí bez jakékoli možnosti mého zásahu. Přátelé nenechávají své přátele používat hudbu na pozadí na svých stránkách, ledaže jsou vaši přátelé z kapely Rush a vy právě pracujete na jejich webových stránkách.

K čemu se JavaScript hodí

Současný web se stále vyvíjí. Jedna z nejpopulárnějších tendencí posledních let je známá jako nevtíravý (unobstrusive) JavaScript. Paradigma nevtíravého skriptování je součást většího hnutí nazvaného *oddělení chování*. Oddělení chování volá po tom, aby byla struktura oddělena od stylů, a obojí od chování. V tomto modelu poskytuje HTML a XHTML strukturu, zatímco CSS poskytuje styly a JavaScript zase chování. JavaScript je nevtíravý; nevstupuje do cesty. Pokud není JavaScript v prohlížeči dostupný, webové stránky stále fungují, protože pro návštěvníka stále existuje jiný způsob, jak webovou stránku používat.

Pokud se použije řádně, nevtíravý JavaScript znamená, že případné selhání při pokusu o použití JavaScriptu se provede s noblesou a grácií. Nepředpokládá se, že JavaScript bude k dispozici – tvůrce nevtíravých skriptů se buď ujistí, že stránka bude bez JavaScriptu fungovat, nebo použije řádné metody zajišťující jeho dostupnost ve chvíli, kdy jej stránka vyžaduje.

Jednu z takových metod si popíšeme v kapitole 14. Nevtíravé skriptování je důležité, neboť pomáhá zajistit přístupnost pro uživatele s alternativními prohlížeči, textovými čtečkami nebo pro jakýkoli další případ.

Já sám jsem zastánce nevtíravého skriptování, protože směřuje k tomu, že se budou dodržovat standardy a výsledné stránky budou dodržovat čtyři doporučení, jež jsem shrnul v předchozí části. Dejte ale tomu, že tohle není ten případ. Člověk může oddělit HTML, CSS a JavaScript a stejně skončit s proprietárními (vlastními) značkami. Když programujete v nevtíravém slova smyslu, máte tendenci dávat si pozor na detaily, a proto se více staráte o to, aby konečný výsledek odpovídal standardům.

V této knize vám ukážu nejen základy JavaScriptu, ale také ten nejlepší způsob, jakým JavaScript používat tak efektivně, jak jen to je možné, tedy nevtíravě.

Poznámka k JScriptu, JavaScriptu a této knize

Tato kniha se věnuje JavaScriptu, jak jej definuje standard ECMA. V některých oblastech zvlášť ještě zmíním podobné jazyky – Microsoft JScript a JScript .NET. Přehledné informace o JScript samotném poskytují následující informační zdroje na Internetu:

JScript 7.1 – Visual Studio 2003 - .NET Framework 1.1: http://msdn2.microsoft.com/en-us/library/72bd815a(VS.71).aspx

JScript 8.0 – Visual Studio 2005 - .NET Framework 2.0: http://msdn2.microsoft.com/en-us/library/72bd815a(vs.80).aspx

Který prohlížeč by webové stránky měly podporovat?

Zpětná kompatibilita je problém webových vývojářů dlouhou dobu. Výběr podpory té které verze prohlížeče se stal kompromisem mezi posledními novinkami v nabídce funkcionality nejnovějších prohlížečů a kompatibilitou požadovanou staršími prohlížeči k prohlížení stránek. Neexistuje žádné jasné a rychlé pravidlo pro to, jaký prohlížeč byste měli na vašich webových stránkách podporovat, takže odpověď zní: to záleží na okolnostech.

Váš výběr závisí na tom, jak budete chtít svoje stránky provést a zda cena návštěv lidí používajících starší hardware a software je více, než hodnota přidané funkcionality nabízené posledními verzemi prohlížečů. Některé prohlížeče jsou ale zkrátka příliš staré, aby je bylo třeba podporovat, neboť například neumí správně vykreslovat CSS, mnohem méně pak JavaScript. Klíčem k podpoře více verzí prohlížečů je otestovat je. Registrace účtu v MSDN od Microsoftu vám poskytne přístup ke starým produktům včetně starších verzí Internet Exploreru, takže se můžete podívat, jak vaše stránky reagují při návštěvě se starším Internet Explorerem 4.

Mnoho webových návrhů (designů) a funkcí JavaScriptu nevyžaduje novější verze webových prohlížečů. Nicméně, jak už jsem řekl, je vždy dobré si ověřit, že se vaše stránky správně vykreslují v různých prohlížečích. I když rozsáhlé testování není možné, minimalizování škod způsobených nezcela správným vykreslením ve starších prohlížečích je taktéž důležité.

Cvičení

1. Správně nebo špatně: JavaScript definují standardy a podporují jej všechny prohlížeče.
2. Správně nebo špatně: Když uživatel s vypnutou podporou JavaScriptu vstoupí na webové stránky, měli byste mu přístup ke stránkám zablokovat, dokud nebude mít dobrý důvod k tomu, aby měl podporu JavaScriptu vypnutou.
3. Vytvořte definiční blok JavaScriptu, který vložte do hlavičky webové stránky (značka `<head>`).
4. Správně nebo špatně: Je důležité deklarovat použitou verzi JavaScriptu v elementu DOCTYPE.
5. Správně nebo špatně: JavaScript lze umístit jak do hlavičky HTML stránky (do značky `<head>`), tak do těla stránky HTML (do značky `<body>`).

Kapitola 2

Programujeme v JavaScriptu

Po přečtení této kapitoly budete schopni:

- Porozumět možnostem, které vývoj v JavaScriptu nabízí
- Nakonfigurovat váš počítač pro vývoj v JavaScriptu
- Používat Microsoft Visual Studio 2008 k vytvoření a nasazení javascriptových aplikací
- Používat Eclipse k vytvoření a nasazení javascriptových aplikací
- Používat Poznámkový blok (nebo jiný editor) k vytvoření javascriptových aplikací
- Porozumět možnostem ladění kódu v JavaScriptu

Možnosti vývoje v JavaScriptu

Protože JavaScript není kompilovaným jazykem, nevyžaduje žádné zvláštní nástroje nebo vývojová prostředí k psaní a vytváření javascriptových aplikací. Taktéž k běhu aplikací v JavaScriptu není zapotřebí žádný zvláštní serverový software. Nicméně vaše možnosti tvorby javascriptových programů jsou virtuálně limitované.

Programový kód v JavaScriptu lze psát v jakémkoli textovém editoru nebo v jakémkoli programu, který používáte k psaní kódu HTML nebo CSS, nebo též ve výkonném integrovaném vývojovém prostředí (Integrated Development Environment – IDE) jako je Visual Studio – někdy můžete využít všech tří možností. Například můžete začít vytvářet svou webovou aplikaci ve Visual Studiu, ale poté zjistíte, že musíte použít textový editor jako je Poznámkový blok k úpravě kódu v JavaScriptu. Ve skutečnosti tedy platí, že byste měli použít takový nástroj, v němž se vám programový kód píše nejlépe.

V této knize probereme, jak provádět vývoj v JavaScriptu za pomoci Visual Studia, ale občas možná doporučím (a ukážu), jak programový kód psát i v textovém editoru, jako je třeba Poznámkový blok nebo Vim (který můžete získat z Internetu z adresy *http://www.vim.org*). Jindy budete moci programový kód jednoduše psát přímo do adresního řádku vašeho webového prohlížeče pomocí pseudoprotokolového identifikátoru *javascript:*, jak už jste viděli v kapitole 1, *JavaScript umí mnohem více, než si myslíte*.

Pokud již nějaký čas vyvíjíte programy v JavaScriptu, možná si všimnete toho, že děláte ty samé věci pro každou webovou stránku. V takových případech je možné kód jednoduše zkopírovat a vložit do další webové stránky, kterou právě vyvíjíte. Nebo ještě lépe byste mohli vytvořit externí soubor obsahující běžné funkce, jež používáte při vývoji stránek stále dokola. Kapitola 10, *Objektový model dokumentu*, poskytuje více informací o funkcích, jež uvidíme v akci v prvních deseti kapitolách.

Konfigurujeme své prostředí

V této části se podíváme na vývoj v JavaScriptu pomocí několika různých nástrojů. Věřím, že budete schopni používat kterýkoli z nástrojů, v němž se vám bude dobře vyvíjet JavaScript a webové stránky, takže toto nepovažujte za vyčerpávající nebo předpojatý seznam nástrojů k vývoji JavaScriptu.