2

JavaScript v jazyku HTML

Zavedení JavaScriptu do webových stránek se okamžitě střetlo s dominantním jazykem webu, kterým je HTML. Společnost Netscape se v rámci své původní práce na jazyku JavaScript snažila vymyslet způsob, jakým by JavaScript mohl koexistovat se stránkami HTML, aniž by narušil jejich vykreslování v ostatních prohlížečích. Prostřednictvím pokusů, omylů a polemik proběhlo nakonec několik jednání, na nichž došlo k dohodě o přidání univerzální podpory skriptování na webové stránky. Většina z toho, co bylo vykonáno v těchto raných dobách webu, přežilo až dosud a dočkalo se své formalizace ve specifikaci jazyka HTML.

Element script

Hlavní metodou pro vkládání JavaScriptu do stránek HTML je element `script`. Ten byl vytvořen společností Netscape, poprvé implementován v prohlížeči Netscape Navigator 2 a později přidán do formální specifikace jazyka HTML. HTML 4.01 definuje pro element `script` následujících pět atributů:

- `charset` (nepovinný): Znaková sada kódu specifikovaného pomocí atributu `src`. Používá se jen zřídka, protože většina prohlížečů jeho hodnotu ignoruje.
- `defer` (nepovinný): Označuje, že provedení skriptu lze bezpečně odložit až do okamžiku, kdy bude obsah dokumentu kompletně analyzován a zobrazen.
- `language` (zastaralý): Původně označovat skriptovací jazyk používaný uvedeným blokem kódu (např. `"JavaScript"`, `"JavaScript1.2"` nebo `"VBScript"`). Většina prohlížečů jej ignoruje a neměl by se používat.
- `src` (nepovinný): Označuje externí soubor, jenž obsahuje kód, který se má provést.
- `type` (povinný): Nahrazuje atribut `language`. Označuje typ obsahu (též nazýván typ MIME) skriptovacího jazyka použitého v uvedeném bloku kódu. Tradičně se jako hodnota uvádí `"text/javascript"`, ačkoliv jak `"text/javascript"`, tak i `"text/ecmascript"` jsou zastaralé. Pro soubory JavaScriptu se obvykle používá typ MIME `"application/x-javascript"`, i když

nastavení této hodnoty v atributu `type` může způsobit ignorování skriptu. Mezi další hodnoty, které fungují v prohlížečích kromě Internet Exploreru, patří `"application/javascript"` a `"application/ecmascript"`. Atribut `type` se kvůli maximální kompatibilitě s prohlížeči i přesto nadále nastavuje na hodnotu `"text/javascript"`.

Element `script` lze používat dvěma způsoby: vložením kódu jazyka JavaScript přímo do stránky nebo začleněním JavaScriptu z externího souboru.

Pro vložení kódu jazyka JavaScript stačí v elementu `script` uvést pouze atribut `type`. Kód JavaScriptu se poté umístí přímo do elementu `script` následujícím způsobem:

```
<script type="text/javascript">
  function sayHi()
  {
    alert("Ahoj!");
  }
</script>
```

Kód JavaScriptu obsažený uvnitř elementu `script` je interpretován odshora dolů. V tomto případě je definice interpretována funkce a uložena do prostředí interpretu. Zbytek obsahu stránky se nenačte ani nezobrazí, dokud se nevyhodnotí veškerý kód v elementu `script`.

Při použití vloženého kódu jazyka JavaScript mějte na paměti, že se ve vašem kódu nesmí nikde vyskytovat řetězec `"</script>"`. Například následující kód způsobí při načítání do prohlížeče chybu:

```
<script type="text/javascript">
  function sayScript()
  {
    alert("</script>");
  }
</script>
```

Kvůli způsobu, jakým se analyzují vložené skripty, vidí prohlížeč řetězec `"</script>"` tak, jako by šlo o uzavírací značku `</script>`. Tento problém lze snadno obejít rozdělením řetězce na dvě části, například takto:

```
<script type="text/javascript">
  function sayScript()
  {
    alert("</scr" + "ipt>");
  }
</script>
```

Díky výše uvedeným změnám je již kód pro prohlížeč přijatelný a nezpůsobí žádné chyby.

Pro začlenění JavaScriptu z externího souboru je nutné použít atribut `src`. Jeho hodnotou je adresa URL ukazující na soubor, který obsahuje kód jazyka JavaScript:

```
<script type="text/javascript" src="example.js"> </script>
```

V tomto příkladu se do stránky načte externí soubor jménem `example.js`. Samotný soubor musí obsahovat pouze kód JavaScriptu, který by se jinak objevil mezi otevírací značkou `<script>` a uzavírací značkou `</script>`. Stejně jako v případě vloženého kódu jazyka JavaScript dojde při interpretování externího souboru k pozastavení zpracování obsahu stránky (určitou chvíli také trvá stažení souboru). V dokumentech XHTML můžete uzavírací značku vynechat:

```
<script type="text/javascript" src="example.js" />
```

Tuto syntaxi byste však neměli používat v dokumentech HTML, protože se jedná o neplatný kód jazyka HTML, který nebude správně zpracován prohlížeči, především pak Internet Explorerem.

> Podle nepsané dohody platí, že externí soubory JavaScriptu mají příponu .js. Nejedná se však o žádný požadavek, protože prohlížeče nekontrolují příponu začleňovaných souborů JavaScriptu. Díky tomu zůstává otevřená možnost generovat kód jazyka JavaScript dynamicky pomocí JSP, PHP nebo jiného skriptovacího jazyka na straně serveru.

Je důležité poznamenat, že element `script` s atributem `src` by již neměl mít mezi značkami `<script>` a `</script>` žádný jiný kód.

Jednou z nejsilnějších a nejkontroverznějších částí elementu `script` je jeho schopnost začleňovat soubory JavaScriptu z vnějších domén. Podobně jako u elementu `img` může být atribut `src` elementu `script` nastaven na úplnou adresu URL, která existuje mimo doménu, v níž se nachází daná stránka HTML:

```
<script type="text/javascript" src="http://www.nekde.cz/soubor.js">
</script>
```

Kód z externí domény bude načten a interpretován, jako by byl součástí stránky, která jej načítá. Díky tomu můžete v případě potřeby využívat soubory JavaScriptu z nejrůznějších domén. Nicméně při přístupu k souborům JavaScriptu umístěným na serveru, který nemáte pod kontrolou, buďte nanejvýš opatrní. Zlomyslný programátor by jej totiž mohl kdykoliv nahradit. Při začleňování souborů JavaScriptu z jiné domény se vždy ujistěte, že tuto doménu vlastníte nebo že ji vlastní důvěryhodný zdroj.

Bez ohledu na způsob začlenění kódu se elementy `script` interpretují v pořadí, ve kterém jsou na stránce uvedeny. Kód prvního elementu `script` se musí celý interpretovat před zahájením interpretace druhého elementu `script`, druhý musí být celý vyhodnocen před třetím, a tak pořád dál.

Umístění značky

Všechny elementy `script` se na stránce tradičně umisťují do elementu `head`, například jako v tomto příkladu:

```
<html>
  <head>
    <title> Ukázková stránka HTML </title>
    <script type="text/javascript" src="example1.js"> </script>
    <script type="text/javascript" src="example2.js"> </script>
  </head>
  <body>
```

```
    <!-- zde je obsah -->
  </body>
</html>
```

Hlavním účelem tohoto formátu bylo udržení odkazů na externí soubory (s kaskádovými styly a JavaScriptem) v téže oblasti. Jenže umístění všech souborů JavaScriptu do elementu `head` znamená, že se musí veškerý kód JavaScriptu stáhnout, analyzovat a interpretovat ještě před zahájením vykreslování stránky (vykreslování začne v okamžiku, kdy prohlížeč obdrží otevírací značku `<body>`). U stránek, které vyžadují velké množství kódu jazyka JavaScript, to může způsobit znatelnou prodlevu ve vykreslování stránky, během níž bude okno prohlížeče úplně prázdné. Z tohoto důvodu začleňují moderní webové aplikace všechny odkazy na JavaScript do elementu `body` až pod obsah stránky, jak ukazuje následující příklad:

```
<html>
  <head>
    <title> Ukázková stránka HTML </title>
  </head>
  <body>
    <!-- zde je obsah -->
    <script type="text/javascript" src="example1.js"> </script>
    <script type="text/javascript" src="example2.js"> </script>
  </body>
</html>
```

Tímto způsobem se před zpracováním kódu jazyka JavaScript vykreslí celá stránka. Uživatel tak má dojem rychlejšího načítání stránky, poněvadž byla redukována doba, kterou musel strávit u prázdného okna prohlížeče.

Odložené spouštění skriptů

Jazyk HTML 4.01 definuje pro element `script` atribut s názvem `defer`. Má za úkol signalizovat, že daný skript při svém provádění nezmění strukturu stránky. To znamená, že jej lze bezpečně spustit po analýze celé stránky. Nastavení atributu `defer` v elementu `script` níže uvedeným způsobem má stejný efekt jako umístění elementu `script` na úplný spodek stránky (jak jsme si ukázali v předchozí části):

```
<html>
  <head>
    <title> Ukázková stránka HTML </title>
    <script type="text/javascript" defer="defer" src="example1.js">
    </script>
    <script type="text/javascript" defer="defer" src="example2.js">
    </script>
  </head>
  <body>
    <!-- zde je obsah -->
  </body>
</html>
```

I když jsou elementy `script` v tomto příkladu umístěny v hlavičce dokumentu, provedou se až poté, co prohlížeč obdrží ukončovací značku `</html>`.

Jediná stinná stránka odloženého spouštění skriptů spočívá v tom, že jeho podpora není napříč všemi prohlížeči úplně běžná. Jedinými předními prohlížeči, které podporují atribut `defer`, jsou Internet Explorer a Firefox. Všechny ostatní prohlížeče jej prostě ignorují a nakládají se skriptem zcela běžným způsobem.

> Více informací o dalších způsobech, jak dosáhnout funkcionality podobné atributu `defer`, se dozvíte v kapitole 12.

Změny v jazyku XHTML

Jazyk XHTML (Extensible Hypertext Markup Language – Rozšiřitelný hypertextový značkovací jazyk) vznikl přeformulováním HTML do podoby aplikace XML. Pravidla pro zápis kódu v jazyku XHTML jsou ve srovnání s jazykem HTML přísnější, což se týká i elementu `script` při vkládání kódu jazyka JavaScript. Ačkoliv je následující blok kódu platný v jazyku HTML, v jazyku XHTML již platný není:

```
<script type="text/javascript">
  function compare(a, b)
  {
    if (a < b) {
      alert("A je menší než B");
    } else if (a > b) {
      alert("A je větší než B");
    } else {
      alert("A se rovná B");
    }
  }
</script>
```

V jazyku HTML má element `script` zvláštní pravidla určující způsob, jakým by měl být analyzován jeho obsah. V jazyku XHTML žádná taková zvláštní pravidla neplatí. To znamená, že se symbol menšítko (<) v příkazu `a < b` interpretuje jako začátek nějaké značky, což způsobí syntaktickou chybu, protože za symbolem menšítka nesmí být mezera.

Tato syntaktická chyba se dá opravit dvěma způsoby. Prvním je nahradit všechny výskyty symbolu menšítko (<) odpovídající entitou HTML (`<`). Výsledný kód pak vypadá následovně:

```
<script type="text/javascript">
  function compare(a, b)
  {
    if (a &lt; b) {
      alert("A je menší než B");
    } else if (a > b) {
      alert("A je větší než B");
    } else {
```

```
      alert("A se rovná B");
    }
  }
</script>
```

Tento kód se nyní na stránce XHTML sice bez problémů spustí, je však hůře čitelný. Naštěstí existuje ještě jedna možnost.

Druhou možností pro opravu kódu podle specifikace jazyka XHTML je obalit kód JavaScriptu do sekce `CData`. V jazyku XHTML (a XML) se sekce `CData` používají pro označení oblastí dokumentu, které obsahují volně formátovaný text, který není nutné analyzovat. Díky tomu je možné v této části použít libovolné znaky, včetně symbolu menšítko, aniž by došlo k syntaktické chybě. Její formát vypadá takto:

```
<script type="text/javascript"> <![CDATA[
  function compare(a, b)
  {
    if (a < b) {
      alert("A je menší než B");
    } else if (a > b) {
      alert("A je větší než B");
    } else {
      alert("A se rovná B");
    }
  }
]]> </script>
```

Problém je nyní vyřešen pro webové prohlížeče, které vyhovují specifikaci jazyka XTHML. Řada prohlížečů však této specifikaci stále nevyhovuje, a sekci tudíž `CData` nepodporují. To lze obejít tak, že značky umístíme do `CData` komentářů jazyka JavaScript:

```
<script type="text/javascript">
//<![CDATA[
  function compare(a, b)
  {
    if (a < b) {
      alert("A je menší než B");
    } else if (a > b) {
      alert("A je větší než B");
    } else {
      alert("A se rovná B");
    }
  }
//]]>
</script>
```

Tento způsob zápisu funguje ve všech moderních prohlížečích. I když jde možná o trošku krkolomný zásah, jedná se o platný kód jazyka XHTML, který je elegantním způsobem kompatibilní s prohlížeči nepodporujícími jazyk XHTML.

Zastaralá syntaxe

Když byl poprvé element `script` představen, označoval odbočku od tradiční analýzy kódu jazyka HTML. Uvnitř tohoto elementu bylo nutné aplikovat zvláštní pravidla, což způsobovalo problémy u prohlížečů, které JavaScript nepodporovaly (týkalo se to především prohlížeče Mosaic). Takové prohlížeče by pak vypsaly obsah elementu `script` na stránku, což by zcela zničilo její vzhled.

Společnost Netscape přišla po spolupráci s tvůrci prohlížeče Mosaic s řešením, které by skrylo vložený kód jazyka JavaScript u prohlížečů, které jej nepodporují. Finální řešení spočívalo v uzavření kódu skriptu do komentáře jazyka HTML, například takto:

```
<script><!--
  function sayHi()
  {
    alert("Ahoj!");
  }
//--></script>
```

Pomocí tohoto formátu mohou prohlížeče jako Mosaic bezpečně ignorovat obsah uvnitř elementu `script`, přičemž prohlížeče podporující JavaScript musejí pomocí tohoto vzoru rozpoznat, že se skutečně jedná o JavaScript, který je nutné analyzovat.

I když tento formát i nadále správně rozpoznávají a interpretují všechny webové prohlížeče, není již nezbytně nutný a neměl by se používat.

Vložený kód a externí soubory

Ačkoliv je možné vkládat kód jazyka JavaScript přímo do souborů HTML, je daleko vhodnější začleňovat maximální množství kódu JavaScriptu pomocí externích souborů. Máme-li na zřeteli, že v souvislosti s tímto postupem neexistují žádná pevně daná pravidla, můžeme argumenty hovořící pro používání externích souborů shrnout do následujících bodů:

- **Udržovatelnost:** Kód jazyka JavaScript, který je rozsypaný po nejrůznějších stránkách HTML, se velmi špatně udržuje. Mnohem snadnější je mít nějaký adresář se všemi soubory JavaScriptu, aby tak vývojáři mohli upravovat kód JavaScriptu nezávisle na značkovacím kódu, v němž se používá.
- **Využití mezipaměti:** Prohlížeče ukládají do mezipaměti všechny externě připojené soubory JavaScript podle určitých nastavení, což znamená, že pokud dvě stránky používají tentýž soubor, pak se tento soubor stáhne pouze jednou, což v důsledku vede k rychlejšímu načítání stránek.
- **Čisté řešení:** Díky začlenění JavaScriptu pomocí externích souborů již není nutné používat dříve zmíněné krkolomné zásahy (v případě XHTML nebo komentářů). Syntaxe pro začlenění externích souborů je stejná pro HTML i XHTML.

Režimy dokumentu

Internet Explorer 5.5 zavedl pomocí deklarace `DOCTYPE` princip režimů dokumentu. Prvními dvěma režimy dokumentu byly *režim quirks* (podivný), který nastavil Internet Explorer tak, aby se choval, jako by šlo o verzi 5 (s několika nestandardními prvky), a *režim standards* (standardní), který nastavil Internet Explorer, aby se choval standardnějším způsobem. I když hlavní rozdíl mezi těmito dvěma režimy souvisí s vykreslování obsahu s ohledem na kaskádové styly, je s nimi spojeno ještě několik vedlejších efektů souvisejících s JavaScriptem. Tyto vedlejší efekty budeme postupně probírat v celé knize.

Vzhledem k tomu, že Internet Explorer jako první zavedl princip režimů dokumentu, ostatní prohlížeče se musely přizpůsobit. Přitom se zrodil třetí režim nazvaný *režim almost standards* (téměř standardní), který obsahoval spoustu prvků režimu standards, ale přitom nebyl tak striktní. Hlavní rozdíl spočívá v zacházení s mezerami kolem obrázků (což je nejzřetelnější při použití obrázků v tabulkách).

Režim quirks je aktivován ve všech prohlížečích prostě tak, že se na začátku dokumentu vynechá deklarace `DOCTYPE`. To ovšem není nejvhodnější, protože režim quirks se napříč různými prohlížeči značně liší a bez krkolomných zásahů není možné dosáhnout žádné úrovně skutečné konzistence mezi prohlížeči.

Režim standards se zapíná při použití jedné z následujících deklarací `DOCTYPE`:

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

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

Režim almost standards spouští deklarace `DOCTYPE` typu `Transitional` a `Frameset`:

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

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

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

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

Protože režim almost standards je tak blízký režimu standards, lidé si jejich odlišností často ani nevšimnou. Když pak mluví o „standardním režimu“, mohou mít na mysli kterýkoli z nich. Také detekce režimu dokumentu (které se budeme věnovat později v této knize) mezi nimi nečiní žádný rozdíl.

Internet Explorer 8 zavádí nový režim dokumentu původně pojmenovaný *režim super standards* (nadstandardní režim). Díky režimu super standards se z Internet Exploreru stává verze prohlížeče nejvíce vyhovující všem standardům. Režim quirks se vykresluje, jako by šlo o Internet Explorer 5, zatímco režim standards používá vykreslovací stroj Internet Exploreru 7. Režim super standards je výchozím režimem dokumentu v Internet Exploreru 8, i když je možné jej vypnout pomocí speciální hodnoty v elementu `meta`:

```
<meta http-equiv="X-UA-Compatible" content="IE=7" />
```

Hodnota `IE` v atributu `content` stanoví, která verze vykreslovacího stroje by se měla použít pro vykreslení dané stránky. Cílem je tedy poskytnout zpětnou kompatibilitu pro weby a stránky, které byly navrženy speciálně pro starší verze Internet Exploreru.

Stejně jako režim almost standards, ani režim super standards se při komunikaci obvykle výrazně neodlišuje od režimu standards. V rámci této knihy budeme termínem režim standards označovat jakýkoliv režim odlišný od režimu quirks.

Element noscript

Předmětem zájmu raných prohlížečů byla elegantní zpětná kompatibilita stánek s prohlížeči, které nepodporovaly JavaScript. Za tímto účelem byl vytvořen element `noscript`, který poskytoval alternativní obsah pro prohlížeče bez JavaScriptu. Tento element může obsahovat libovolné elementy HTML (tedy až na element `script`), které lze umístit do těla dokumentu (element `body`). Veškerý obsah v elementu `noscript` bude zobrazen pouze v následujících dvou případech:

- Prohlížeč nepodporuje skriptování.
- Podpora skriptování je v prohlížeči vypnuta.

Pokud nastane některá z výše uvedených podmínek, vykreslí se obsah uvnitř elementu `noscript`. Ve všech ostatních případech se obsah elementu `noscript` nevykreslí.

Níže je uveden jednoduchý příklad:

```
<html>
  <head>
    <title> Ukázková stránka HTML </title>
    <script type="text/javascript" defer="defer" src="example1.js">
    </script>
    <script type="text/javascript" defer="defer" src="example2.js">
    </script>
  </head>
  <body>
    <noscript>
      <p> Tato stránka vyžaduje prohlížeč s podporou JavaScriptu. </p>
    </noscript>
```

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

V tomto příkladu se uživateli zobrazí zpráva, je-li skriptování nedostupné. U prohlížečů podporujících skriptován, se tato zpráva nikdy nezobrazí, ačkoli zůstane součástí stránky.

Shrnutí

Kód jazyka JavaScript se vkládá do stránek HTML pomocí elementu `script`. Pomocí něho lze vložit JavaScript do stránky HTML mezi ostatní značkovací kód nebo vložit JavaScript existující v nějakém externím souboru. Podstatné jsou následující body:

- U obou způsobů použití je nutné nastavit atribut `type` na hodnotu `"text/javascript"`, což signalizuje, že skriptovacím jazykem je JavaScript.
- Pro začlenění externích souborů JavaScriptu je nutné nastavit atribut `src` na adresu URL začleňovaného souboru. Může jít o soubor na tomtéž serveru jako původní stránka nebo o soubor existující na úplně jiné doméně.
- Všechny elementy `script` se interpretují v pořadí, v jakém jsou uvedeny na stránce. Kód obsažený uvnitř elementu `script` se celý interpretuje, než se přejde k dalšímu elementu `script`.
- Prohlížeč musí dokončit interpretaci kódu uvnitř elementu `script` ještě předtím, než může pokračovat ve vykreslování stránky. Z tohoto důvodu se elementy `script` obvykle umisťují ke konci stránky, pod hlavní obsah a před uzavírací značku `</body>`.
- V Internet Exploreru je možné pomocí atributu `defer` odložit provádění skriptu až do okamžiku po vykreslení dokumentu. I když je tento atribut součástí specifikace HTML 4.01, Internet Explorer je jediným prohlížečem, který má implementovanou jeho podporu.

Při použití elementu `noscript` můžete specifikovat obsah, který se má zobrazit pouze tehdy, není-li skriptování v prohlížeči dostupné. Je-li skriptování v prohlížeči zapnuté, žádný obsah umístěný uvnitř elementu `noscript` se nezobrazí.