

JScript a DOM

Dynamické webové stránky

Tomáš Matoušek

tmd.havit.cz

Použití

Výhody:

- Snížení režie při manipulaci s daty
- Přenesení části režie na klienta (uvolnění serveru)
- Zpřehlednění struktury stránek (menu apod.)
- Vylepšení uživatelského rozhraní

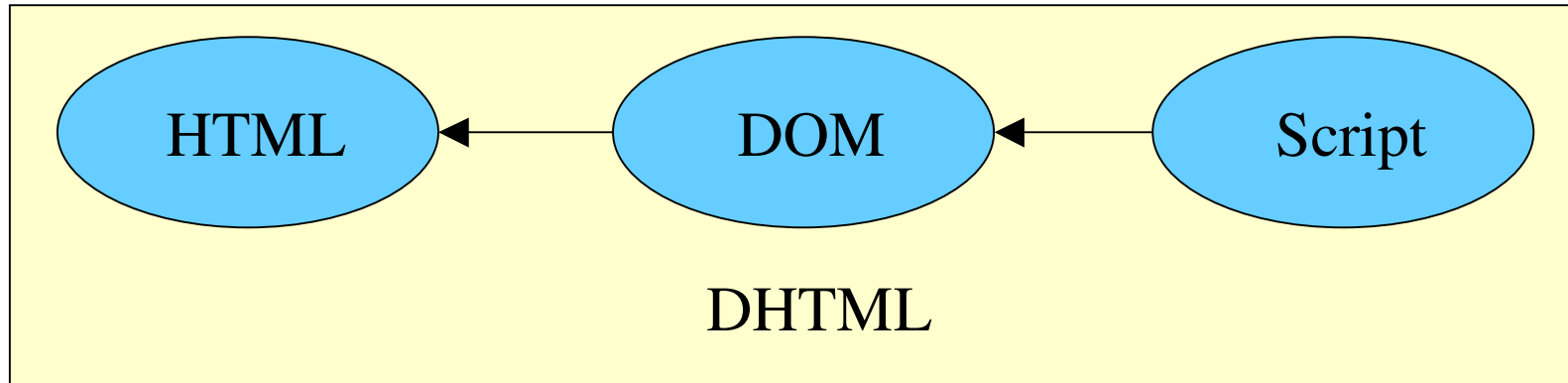
Nevýhody:

- Zvýšené požadavky na funkčnost klienta (musí umět JS)
- Nekompatibilita různých klientů (záleží na cíli řešení)
- Zatížení klienta (většinou však nepatrné)
- Možnost zneužití (zneužít lze ale téměř cokoliv)

Jak vytvářet a měnit HTML dokument?

1. Staticky, tj. při návrhu
2. Dynamicky
 - a) na straně serveru (PHP, ASP, ...)
 - b) na straně klienta (DHTML)

DHTML



Pomocí HTML definujeme strukturu a vlastnosti dokumentu, které klient zobrazí uživateli.

Skript (interpretovaný jazyk) je vykonáván klientem a přes rozhraní DOM (Document Object Model) dokument mění a reaguje na pokyny uživatele.

Skriptem může být JScript, JavaScript, VBScript, ...

JScript, JavaScript, ECMAScript

Jazyk (JavaScript) byl vymyšlen ve firmě Netscape a implementován v prohlížečích Navigator od verze 2.

Převzal ho Microsoft a implementoval ho jako JScript v Internet Exploreru od verze 3.0.

Poté byl standardizován jako ECMAScript.

Nyní je standardem ECMAScript level 3 a stále se vylepšuje.

Tento standard je zahrnut v JavaScriptu 1.3
i v JScriptu 5.5.

Proměnné a datové typy

```
/* Deklarace promennych */  
var i,j = 5,I = 1; //i je nyní neinicializovana, tj. je undefined  
  
var k = null; //k je inicializovana, ale nema hodnotu (null)  
  
var s = 'Cena: ' + 1285.50 + j + ' Kč';  
  
var types = "Typ i je "+typeof(i)+" , typ s je "+typeof(s)+".";  
  
var c = true || false;  
  
d = 10; //implicitní deklarace
```

Primitivní typy: **boolean, number, string, undefined, null**

Složené typy: **object, array, ...**

Loosly-typed language (automatické přetypování, nedefinování typu, ...)

Funkce

```
/* Deklarujeme funkci se tremi parametry */  
function f(a,b,c)  
{ var i = 10;      //deklarujeme lokalni promennou  
  sum = a+b+c;    //implicitní globální deklarace  
  return i+sum;  //návratová hodnota  
}  
  
f(5,5,typeof(f)); //Zavolame f, treti parametr je "function"  
  
function call_function(g)  
{ return g(10);  
}  
  
call_function(function (a) {return a+1;});
```

Objekty

Objekt je seznam pojmenovaných vlastností (properties).

Property může být

- a) hodnota primitivního typu
- b) objekt
- c) funkce (metoda)

Objekt se chová jako asociativní pole obsahující dvojice (název, hodnota).

Lze proto dynamicky přidávat nové property a metody.

Prototypová dědičnost

- pomocí property **prototype**, která ukazuje na předka
- vytvoří se „prototyp“ objektu, který vidí všichni potomci

Vestavěné objekty (intrinsic/built-in objects)

Object	předchůdce všech objektů, základní vlastnosti objektů
Boolean	obalení triviálního typu boolean
Number	obalení triviálního typu number
Date	uchovávání, formátování a práce s datem a časem
Array	asociativní či indexované pole
String	řetězec
Function	funkce (obsahuje metody pro práci s argumenty)
RegExp	regulární výrazy, pattern matching
Global	zpřístupňuje globální konstanty, metody a objekty
Math	matematické konstanty a funkce
a další ...	

Host objects

- objekty přidané hostující aplikací (např. objekty DOM)
- jsou zařazeny jako property objektu **Global**
- budou zmíněny dále, v části věnované DOM

User-defined objects

- lze definovat vlastní objekty
- lze také přidat nové vlastnosti a metody k existujícím objektům

Příklad

```
var A = new Array(function(a) {return a+1;});  
var B = new Array(Math.sin(Math.PI/4), "X", new Date(2002, 12, 2));  
  
A.push("Z"); //přidáme na konec A prvek "Z"  
A[2] = B[1]; //přidáme na konec prvek B[1] (tj. "X")  
  
var s = "Toto je řetězec.".replace(/o/g, "0"); //Regulární výraz  
var C = [3, 2, 4, 1].sort(); //Vytvori pole a usporada je  
var Assoc = {"hello":5, "bye":6}; //Vytvoří asociativní pole  
Assoc.hello = Assoc["hello"] + 1;
```

Objekty vyvážíme pomocí operátoru **new**, kterým voláme *konstruktor* objektu (např. `Array`), což je funkce, která inicializuje objekt.

Některé objekty lze vytvářet i speciálními konstrukcemi (např. **Array** pomocí `[]` či `{}`, **String** `""`, **RegExp** `/ /`, **Function** `function(){}` apod.)

Řídící struktury

```
/* podmínky */  
if (a>0) {} else {}  
switch (a)  
{ case 1:/* kód */;break;  
  default /* kód */  
}  
  
/* cykly */  
for (var q=0;q<50;q++) {}  
for (p in some_array_or_object) {} //toto v C není  
do {} while (true)  
while (i!=1) {}  
  
/* výjimky */  
try {} catch (error_object) {}  
  
/* with */  
with (objekt) {} //toto v C není
```

DOM

- struktura objektů reprezentující stránku, prohlížeč, ...
- z JScriptu technicky dosažitelná pomocí objektu **Global**
- doporučením W3C je HTML 4.0 a DOM level 2
- dále se budeme zabývat DOM v Internet Exploreru

Co je přístupné JScriptu přes DOM?

- všechny HTML tagy (reprezentovány jako objekty)
- jejich atributy včetně CSS (vlastnosti těchto objektů)
- možnost reakce na události vyvolané uživatelem
- vlastnosti klienta (prohlížeče)

Hierarchie objektů (top-level)

window	okno exploreru nebo frame
— document	dokument HTML
— event	uchovává informace o událostech
— history	pole naposledy navštívených dokumentů
— navigator	informace o prohlížeči
— screen	vlastnosti o obrazovky (rozlišení, ...)
— status	stavová řádka

Nejvýše v hierarchii stojí objekt **window**. Pro zkrácení zápisů se v tečkové notaci nemusí psát. Např. **window.event** odpovídá **event**.

Hierarchie objektů (další úrovně)

Objekt **document** již reprezentuje vlastní dokument popsaný HTML včetně vložených skriptů, stylů, atd. Celá hierarchie tvoří strom tagů tak, jak jsou vnořovány do sebe pomocí HTML.

K tagům se přistupuje pomocí kolekcí.

Kolekce se chová jako pole obsahující reference na vybrané tagy.

K tagům v kolekci můžeme vždy přistupovat pomocí indexů:

```
kolekce[i] //kolekce jako pole (tag: i-tá položka)
```

```
kolekce(i) //kolekce jako funkce vracející i-tý tag
```

Je-li tag pojmenován (tj. má-li nastaven atribut **id** nebo **name**), lze místo indexu použít toto jméno. Z vlastnosti asociativních polí plyne možnost použití zápisu:

```
kolekce.id_objektu; //ekvivalentní s kolekce[id_objektu]
```

Předdefinované kolekce

objekt **document**:

all všechny tagy v dokumentu
images obrázky (tagy IMG)
forms formuláře (tagy FORM)
a další ...

každý objekt reprezentující tag:

all všechny vnořené tagy
children vnořené tagy na první úrovni vnoření

Navíc má každá kolekce tagů metodu **tags**, která vrací kolekci tagů daného druhu.

```
document.all.tags("H1"); //vrací kolekci nadpisů  
document.all.tags("IMAGE"); //odpovídá kolekci images
```

A to je vše z této přednášky.

Další informace

JScript msdn.microsoft.com

[/library/en-us/script56/html/js56jslrfJScriptLanguageReference.asp](http://msdn.microsoft.com/library/en-us/script56/html/js56jslrfJScriptLanguageReference.asp)

JavaScript developer.netscape.com

[/docs/manuals/js/core/jsref/index.htm](http://developer.netscape.com/docs/manuals/js/core/jsref/index.htm)

ECMAScript <ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf>

IE DOM msdn.microsoft.com

[/workshop/author/dhtml/reference/dhtml_reference_entry.asp](http://msdn.microsoft.com/workshop/author/dhtml/reference/dhtml_reference_entry.asp)

W3C DOM www.w3.org/DOM/DOMTR