

JavaScript v Seznamu

Michal Aichinger



Agenda

- Seznámení s Javascriptem
- Objektové programování v Javascriptu
- Naslouchání událostem v Javascriptu
- Knihovna JAK
- Widgety postavené nad JAKem
- Nevtrivní (Unobtrusive) Javascript
- Testování Javascriptu

Seznámení s Javascriptem

Co je Javascript

- Scriptovací jazyk
- Prototypový jazyk
- Objektový jazyk

Co není Javascript

- Třídní jazyk
- Java

Proč používat Javascript

Odlehčení serveru:

- Výpočetní čas
- Přenosová kapacita

Zlepšení uživatelského prožitku z používání aplikace

JavaScript a Seznam



JAK ...?

Javascript a Seznam = JAK

Knihovna JAK

<http://jak.seznam.cz>

Další opensource projekty: [CAPTCHA API](#) | [CFG.Parser](#) | [DRG.Log](#) | [FastRPC](#) | [FastRPC-netcat](#) | [KJS.Compress](#) | [Seznam.FS](#) | [TENG](#)



JAK
JavaScriptová knihovna

Úvod [Příklady](#) [Stáhnout](#) [Manuál](#)

Úvod

JAK je kompaktní a jednoduchý systém volně provázaných knihoven, usnadňující práci v prostředí jazyka JavaScript.

Vznikl na základě potřeby standardizovat a zjednodušit práci při vývoji JS aplikací tak, aby poskytli nástroje pro řešení často se opakujících požadavků.

JAK byl napsán s ohledem na tyto cíle:

1. Nevytvářet zbytečně složité konstrukce a pravidla
2. Co nejméně modifikovat „běžové prostředí“ JavaScriptu
3. Zapouzdřit vlastní funkcionalitu, tj. používat vlastní „namespace“

JAK řeší:

1. detekci klienta a prostředí
2. práci s událostmi
3. práci s DOM a HTML (nejčastěji používané postupy)
4. práci s XMLHttpRequest (AJAX)

A navíc přidává pokročilé nástroje pro OOP:

1. vytváření tříd
2. jednoduchou implementaci dědičnosti
3. vytváření složitěji strukturovaných aplikací
4. vytváření a zpracovávání vlastních událostí

JAK je kompletně zdokumentovaný pomocí [JSDoc Toolkitu](#). Kromě dokumentace lze pro seznámení s JAK a jeho jednotlivými částmi použít i příklady použití.

JAK se vyčlenil z projektu [Mapy.cz](#) a nyní se používá i na dalších službách Seznamu: [Homepage](#), [Týpogram](#) a [Fotomapy](#).

JAK podporuje následující moderní prohlížeče: Internet Explorer verze 6 a výš, FireFox 1.5 a výš (a prohlížeče, které používají stejné Gecko jádro), Safari 3, Opera 8.5 a výš a Konqueror 3.5.5.

Novinky

1.9.2009
Opravena zabalená verze knihovny JAK. Dále byl přidán nový widget [Slider](#).

11.8.2009
V [LightBoxu](#) opravena metoda `SZ.N.LightBox.create()`, která nyní načítá popisky obrázků z jejich altů.

23.7.2009
Ve [sloupcovém grafu](#) lze nyní kreslit data posunutá od začátku i konce ostatních řad. Dále byl přidán na stránku [konzole](#) bookmarklet, pro použití konzole na libovolné stránce.

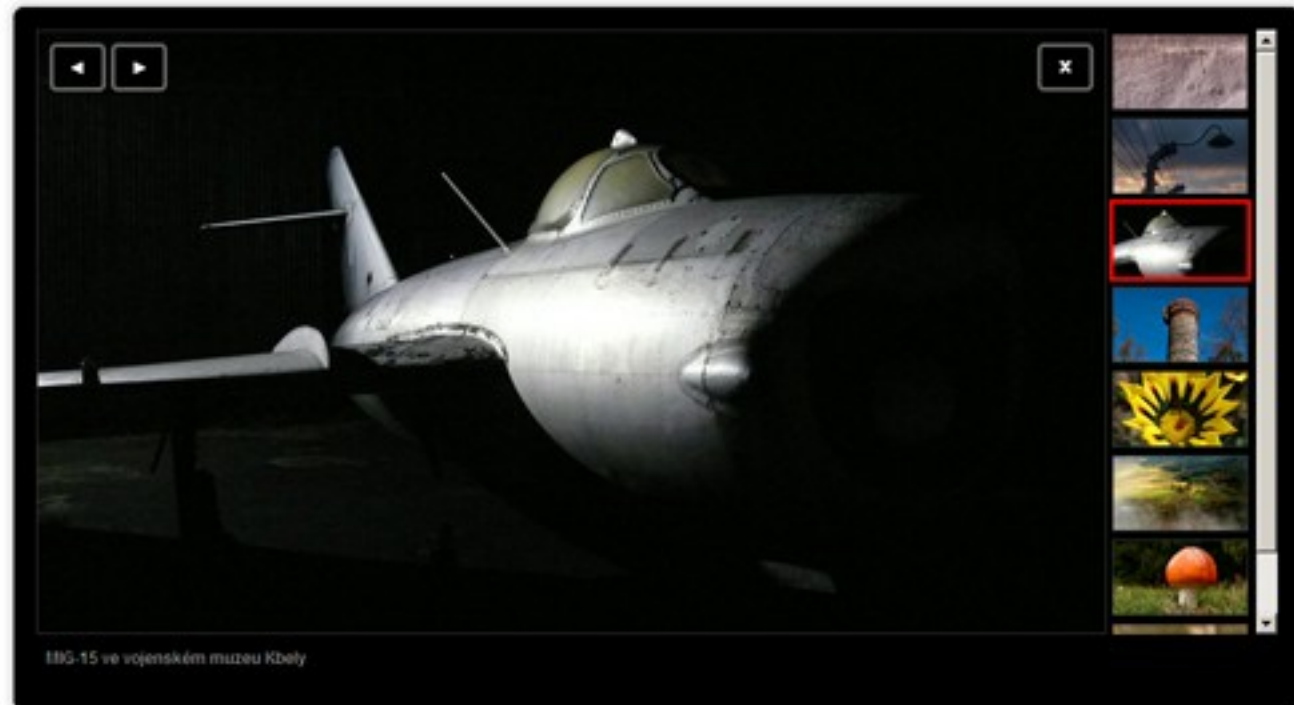
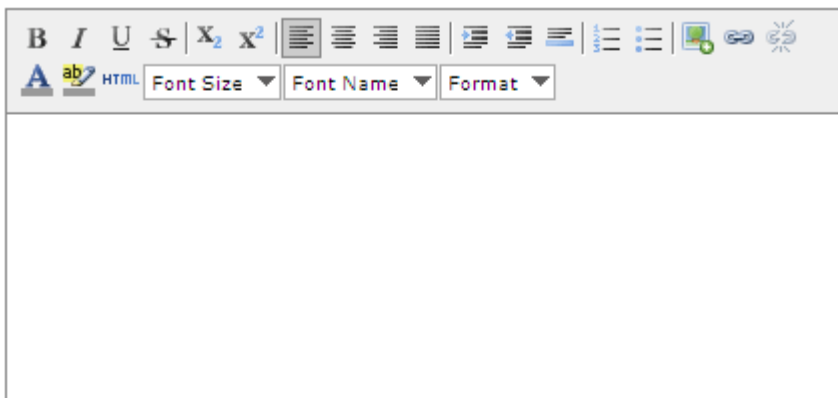
3.6.2009
V prototypu Date změněny výchozí názvy dnů a měsíců na české. [Kalendář](#) nyní využívá tyto řetězce. Vylepšeny [widgety](#) pro kreslení grafů.

23.4.2009
Přidán widget [LightBox](#) - nová konfigurovatelná a skinovatelná prohlížečka obrázků.

Knihovna JAK

Knihovna obsahuje:

- Core (základní) moduly zapouzdřující práci s DOM, událostmi, XMLHttpRequestem
- Widgety, stavějící nad základní funkcionalitou, např.: WYSIWIG Editor, galerie Lightbox, kalendář, slider, ...



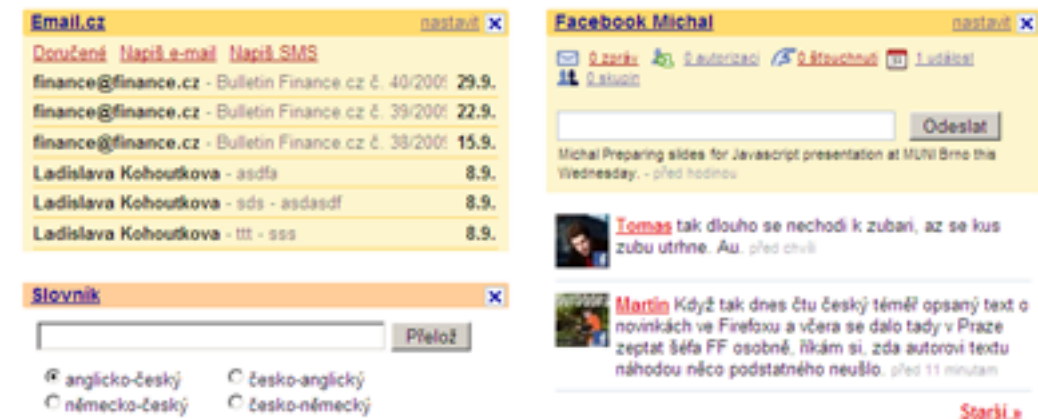
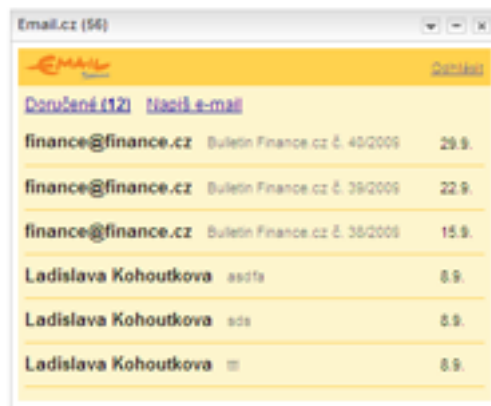
Kde Seznam využívá Javascript

- webový prohlížeč - od oživení webových stránek, až po aplikace
- Firefox – plugin Lištička (ve výrobě pro 3.5)



- Gadgets – spolupráce s třetími stranami (Facebook aplikace pro HP, Email a Lidé pro Vodafone)

Vodafone park beta
Development Console



Programování v Javascriptu

- Lze programovat jak funkcionálně tak i objektově
- Objektový přístup je z pohledu rozšiřitelnosti a údržby lepší
- Javascript pro vnitřní funkce setTimeout, setInterval a addEventListener očekává pouze callback funkci a ne metodu objektu

Objektové programování

Vytváření objektů pomocí literálu:

```
var mujObj = {  
    hodnota_1 : "test",  
    metoda_1 : function() { return this.hodnota_1; }  
};
```

```
mujObj.metoda_1();
```

Objektové programování

Vytváření objektů pomocí new Object:

```
var mujObj = new Object();  
mujObj.hodnota_1 = "test";  
mujObj.metoda_1 = function() {  
    return this.hodnota_1;  
};
```

```
mujObj.metoda_1();
```

Objektové programování

Vytváření objektů pomocí konstrukční funkce a new:

```
var mujConstructor = function() {  
    this.hodnota_1 = "test";  
    this.metoda_1 = function() {  
        return this.hodnota_1;  
    };  
};  
  
var mujObj = new mujConstructor();  
mujObj.metoda_1();
```

Objektové programování

Vytváření objektů pomocí konstrukční funkce, prototypů a new:

```
var mujConstructor = function() {  
    this.hodnota_1 = "test";  
};  
mujConstructor.prototype.m_1 = function() {  
    return this.hodnota_1;  
};
```

```
var mujObj = new mujConstructor();  
mujObj.m_1();
```

Objektové programování s JAKem

Modul ClassMaker v JAKu umožňuje vytvářet „class-like“ zápis kódu s možností dědění a implementace rozhraní

Objektové programování s JAKem

```
var Vehicle = SZN.ClassMaker.makeClass ({  
  NAME: "Vehicle",  
  VERSION: "1.0",  
  CLASS: "class"  
});  
//konstruktor s definici vlastnosti instanci  
Vehicle.prototype.$constructor = function() {  
  this.passengers = 0;  
};  
//definice metody  
Vehicle.prototype.addPassenger = function() {  
  this.passengers++;  
}
```

Objektové programování s JAKem

```
var Car = SZN.ClassMaker.makeClass ({
  NAME: "Car",
  VERSION: "1.0",
  CLASS: "class",
  EXTEND: Vehicle
});
//konstruktor s definici vlastnosti instanci
Car.prototype.$constructor = function() {
  this.$super();
  this.engineOn = false;
};
//definice metody
Car.prototype.startEngine = function(){
  this.engineOn = true;
}
```


Objektové programování s JAKem

Implementace rozhraní kopíruje reference na metody definované v rozhraní

```
var Car = SZN.ClassMaker.makeClass ({  
    NAME: "Car",  
    VERSION: "1.0",  
    CLASS: "class",  
    EXTEND: Vehicle,  
    IMPLEMENT: [IMovable, ICountable]  
});  
  
...  
var car = new Car();
```

Událostní programování

Javascript umožňuje zpracovat události vzniklé interakcí uživatele a stránky.

- Zpracování je asynchronní
- Pro navěšení funkčnosti na událost se využívá metody `addEventListener`
- Odvěšení posluchače pomocí metody `removeEventListener`, kdy si musíme pamatovat všechny parametry zadané metodě `addEventListener`

Událostní programování

Příklad s navěšením ovladače na klik:

```
<a href="#" id="odkaz">Odkaz</a>
<script type="text/javascript">
function click_func (e) {
    alert("bylo kliknuto");
}

var elm = document.getElementById("odkaz");
elm.addEventListener("click", click_func);

...
elm.removeEventListener("click", click_func);
</script>
```

Objektový programování a události

Při použití objektů narazíme na omezení, kdy nejde předat instance objektu a její metoda

```
<script type="text/javascript">
function _bind(obj, fnc) {return function(){
    return fnc.apply(obj, arguments);
}}

var car = { name: "auto",
            click_func : function(e) {alert(this.name);}
}
car.click_func = _bind(car,car.click_func);

elm.addEventListener("click", car.click_func);
</script>
```

Událostní programování v JAKu

```
<a href="#" id="odkaz">Odkaz</a>
<script type="text/javascript">
Car = SZN.ClassMaker.makeClass({NAME:"Car", CLASS:"class"});

Car.prototype.click_func = function (e, elm) {
    alert("bylo kliknuto na " + e.href);
}

var car = new Car();
var elm = SZN.gEl("odkaz");
var id = SZN.Events.addListener( elm, "click", car,
    click_func);
...
SZN.Events.removeListener(id);
</script>
```

Další moduly v knihovně JAK

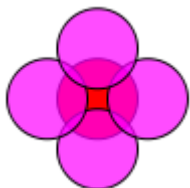
- **Browser** – detekce klienta a jeho verze
- **ClassMaker** – tvorba „tříd“
- **Dom** – metody pro práci s DOM stromem stránky
- **Events** – zapouzdření metod pracujících s událostmi nad elementy v DOM stromu
- **Request** – objekt pro práci s XMLHttpRequestem a AJAXovým dotazováním serveru
- **Signals** – tvorba uživatelských událostí pomocí vzoru Observer a jejich zpracování
- **Components** – zapouzdření metod pro vytváření komponent a jejich skládání

Widgets v JAKu

- Kalendář
- Hodnotící prvek
- Editor
- Vyskakovací okno
- Lightbox
- Ořez obrázků
- Záložky
- Výběr barvy
- Výběr
- Řazení
- Konzole
- Slider

Animace a grafika v JAKu

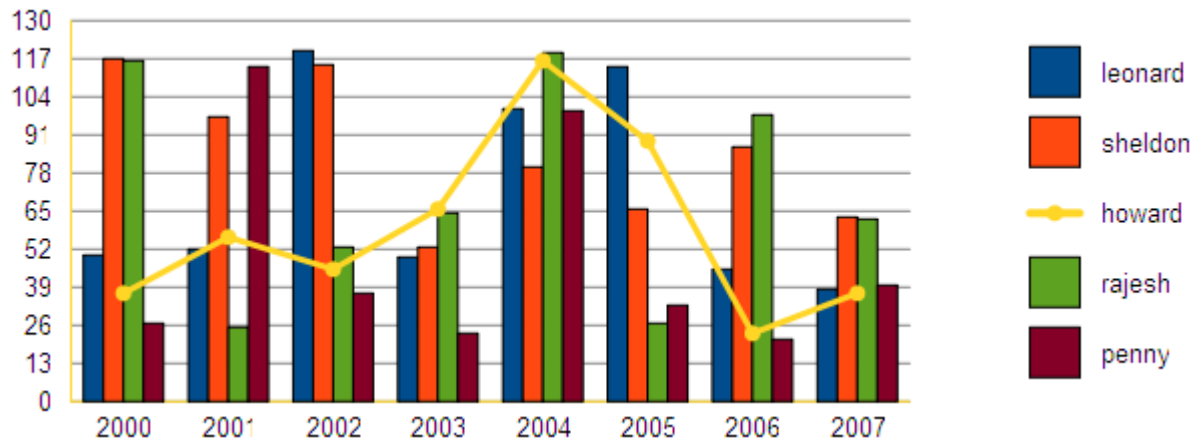
- JAK sjednocuje práci s vektory (SVG a VML)



- Pro animace a složitější efekty je připravena třída Interpolator s možností volby interpolační křivky
- Nad tímto Interpolátorem staví třída CSSInterpolator, pomocí které lze přímo animovat CSS vlastnosti
- Výhodou použití CSSInterpolator je použití pouze jednoho časovače pro změnu více vlastností

Grafy v JAKu

Nad vektorovou knihovnou jsou napsány knihovny pro zobrazení koláčového, sloupcového a spojnicového grafu



Nevtíravý (Unobtrusive) Javascript

7 principů jak se jako vývojář chovat ke zdrojovému kódu stránky a k uživateli – oddělení obsahu od chování

- Nevytvářejte si mylné předpoklady - v prohlížeči uživatele nemusí JS fungovat, nebo být dostupná požadovaná funkčnost
- Vztah mezi HTML a JS - jak propojit funkčnost a vzhled
- Nechte traverzování stromem na expertech - měňte radši CSS třídy
- Nutnost porozumět uživateli a jeho prohlížeči
- Porozumět Javascriptovým událostem a jejich bublání
- Myslet na ostatní - zbytečně neplnit globální prostor
- Tvořte pro další vývojáře - údržba aplikace bude snazší

Nevtíravý (Unobtrusive) Javascript

Test funkčnosti a detekce prohlížeče obecně:

```
if (window.XMLHttpRequest) { /*pouzij request*/ }  
if (window.opera) { /*uzivatel pouziva operu*/ }
```

Test funkčnosti se v JAKu provádí skrytě za účelem určení typu prohlížeče a jeho verze:

```
alert (SZN.Browser.client);  
alert (SZN.Browser.version);
```

Nevtíravý (Unobtrusive) Javascript

Propojení funkčnosti a vzhledu špatně:

```
<a href="#" onclick="udelej(this)">Odkaz</a>
```

Správné řešení je mít veškerý kód JS v externím souboru a navěsit ovladače pomocí JS.

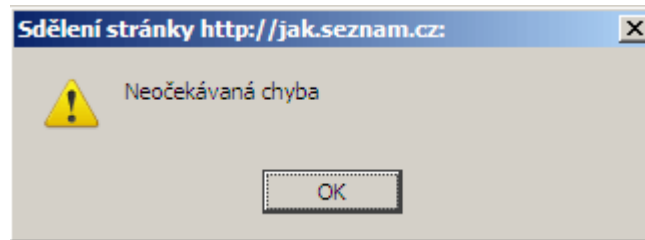
Problém je s prodlevou mezi plným načtením stránky a jejím zobrazením, většinou používáme:

```
<a href="#" id="odkaz">Odkaz</a>
```

```
<script>SZN.Events.addListener(SZN.gEl("odkaz"), "click",  
obj, "udelej");</script>
```

Nevtíravý (Unobtrusive) Javascript

Chybové hlášky uživateli příliš v práci nepomohou, reagovat by měla aplikace.



Testování Javascriptu

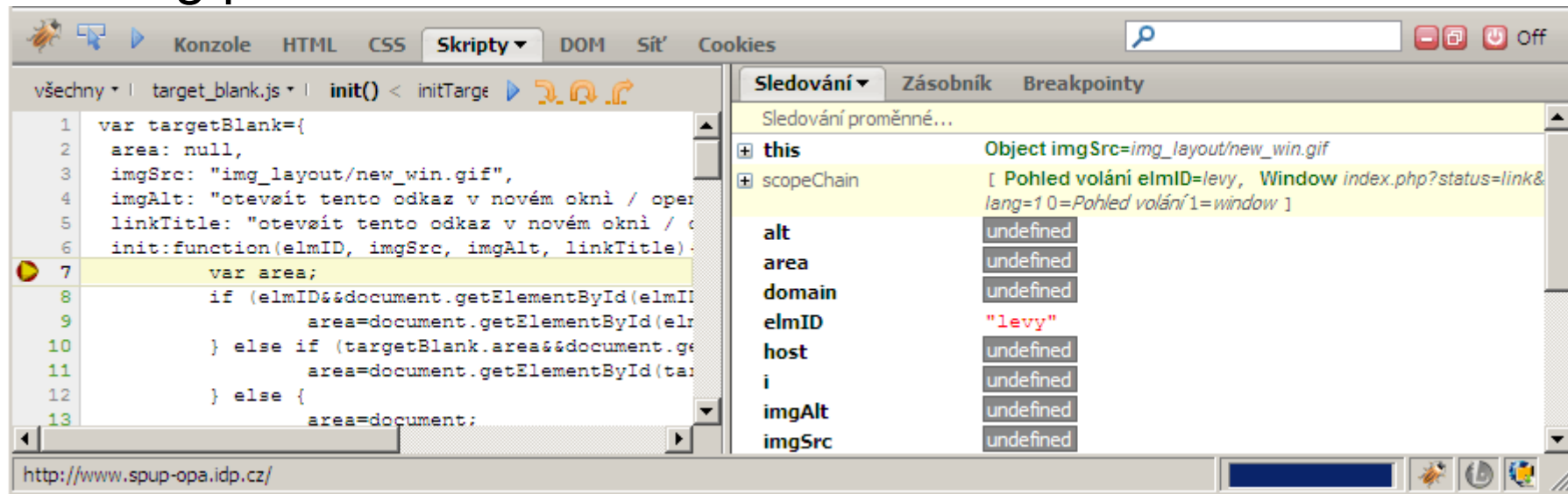
Dva typy programátorů:

- Programátor, který píše bez chyb
- Programátor, který si myslí, že píše bez chyb

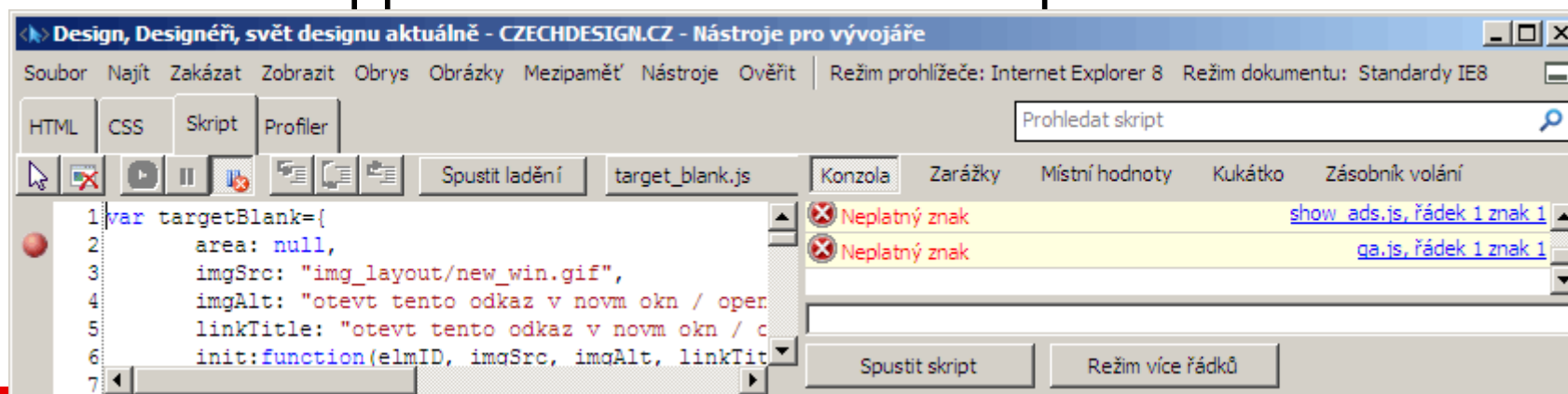
První je nedosažitelný ideál, druhý, je každý z nás, oba by měli používat ladící nástroje.

Testování Javascriptu

- Firebug pro Firefox



- Web Developer Toolbar v Internet Exploreru 8



Shrnutí

- Knihovna JAK je zdarma pro nekomerční i komerční použití
- JAK je vydán pod licencí MIT
- Domovská stránka <http://jak.seznam.cz>

Děkuji za pozornost

Michal Aichinger

michal.aichinger@firma.seznam.cz