Kapitola 8

Objekty v JavaScriptu

Po přečtení této kapitoly budete schopni:

- Porozumět objektům v JavaScriptu, včetně atributů objektů, metod objektů a tříd
- Vytvářet objekty
- Definovat atributy objektů
- Porozumět polím v JavaScriptu
- Použít několik metod objektu `Array`
- Porozumět objektu `Date` JavaScriptu
- Použít objekt `Date` pro zobrazení aktuálního data ve webové stránce

Objektově orientované programování

Ti z vás, kteří se ještě nesetkali s pojmem objektově orientované programování nebo potřebují drobné osvěžení paměti, pokračujte dále ve čtení. Pokud objektově orientované programování již znáte, přeskočte dále k části s názvem *Vytváření objektů*.

Paradigma programování popisuje metodologii pro řešení problémů. Existuje více jak 25 různých paradigmat programování, z nichž některé v dnešních programech budete jen stěží hledat. O jiných jste mohli slyšet nebo je dokonce použít, aniž byste o tom věděli. Existuje např. funkcionální programování, programování řízené událostmi, programování orientované na komponenty, strukturované programování a mnoho dalších.

Paradigmata programování přichází a zase odchází. Objektově orientované programování je však známo už mnoho let a zatím to nevypadá, že by v dohledné době mělo vymizet. Pokud jste se tedy základy objektově orientovaného programování ještě nenaučili, není vhodnější okamžik než nyní.

Tato část nemůže probrat více, než jen nezbytné základy této problematiky. Chci vás seznámit především s objektově orientovanými technologiemi, abyste znali postupy používané při objektově orientovaném programování a terminologii často používanou programátory v JavaScriptu.

Objekty

Objekty jsou věci. V reálném světě, na rozdíl od toho virtuálního a abstraktního světa programování, jsou míč, stůl nebo auto objekty. Objekt je něco s popsatelnými vlastnostmi, s čím se dá pracovat a co se chová určitým způsobem. Objekt v paradigmatu objektově orientovaného programování je kombinací programového kódu a dat, který podobně vykazuje vlastnosti a určité chování.

Atributy

Objekt má atributy. Budeme-li opět hledat analogii ve skutečném světě, pak např. objekt míč může mít atribut barva – s hodnotou např. červená, bílá nebo vícebarevný. Může mít také atri-

but velikost – může se jednat o malý míček, např. na baseball, anebo o větší na basketbal, popř. o úplně jiný. Tyto atributy je možné reprezentovat jako:

```
mic.barva
mic.velikost
```

Metody

Stejně jako mohou mít objekty atributy, mohou mít i metody. Metody definují způsob, jakým se objekt chová. Míč může mít metodu kutálet se, která by určila jak daleko se míč dokutálí. Teoreticky ne všechny objekty mají metody a ne všechny objekty mají atributy, ačkoli v praxi všechny objekty mají přinejmenším jednu metodu nebo atribut.

Z kapitoly 7, *Práce s funkcemi*, víte, že metoda je pouze funkce patřící určitému objektu. Definice metody pro kutálení se míče používající anonymní funkci může vypadat následovně:

```
mic.kutaletSe = function() {
  var vzdalenost = this.velikost * this.aplikovanaSila;
}
```

> **Co znamená this?**
>
> Předchozí příklad ukázal použití něčeho nového, klíčového slova `this`. Toto klíčové slovo se používá pro odkázání se na aktuální objekt, tj. objekt, kterému aktuální atribut nebo metoda patří. V kontextu objektů klíčové slovo `this` odkazuje na volající objekt. Klíčové slovo `this` je možné použít pro nastavení hodnoty atributu objektu v těle jeho metody.
>
> Klíčové slovo `this`, jak uvidíte v kapitole 11, *Použití JavaScriptu ve webových formulářích*, je velkým pomocníkem pro vývojáře v JavaScriptu při validaci obsahu webových formulářů.

Třídy

Třídy definují kolekce objektů, které sdílí stejné atributy a metody. Třídy zjednodušují vytváření více objektů stejného typu. Uvažme následující příklad. V předchozích kapitolách jsem v některých příkladech použil objekt `hvezda`. Výpis 8.1 ukazuje, kolik programového kódu by bylo zapotřebí pro webovou stránku s informacemi o 14 důležitých hvězdách.

Výpis 8.1. Vytvoření objektu hvězdy

```
var hvezda = {};

hvezda["Polaris"] = new Object;
hvezda["Mizar"] = new Object;
hvezda["Aldebaran"] = new Object;
hvezda["Rigel"] = new Object;
hvezda["Castor"] = new Object;
hvezda["Albireo"] = new Object;
hvezda["Acrux"] = new Object;
hvezda["Gemma"] = new Object;
hvezda["Procyon"] = new Object;
hvezda["Sirius"] = new Object;
hvezda["Rigil Kentaurus"] = new Object;
hvezda["Deneb"] = new Object;
hvezda["Vega"] = new Object;
hvezda["Altair"] = new Object;

hvezda["Polaris"].souhvezdi = "Ursa Minor";
hvezda["Mizar"].souhvezdi = "Ursa Major";
```

```
hvezda["Aldebaran"].souhvezdi = "Taurus";
hvezda["Rigel"].souhvezdi = "Orion";
hvezda["Castor"].souhvezdi = "Gemini";
hvezda["Albireo"].souhvezdi = "Cygnus";
hvezda["Acrux"].souhvezdi = "Crux";
hvezda["Gemma"].souhvezdi = "Corona Borealis";
hvezda["Procyon"].souhvezdi = "Canis Minor";
hvezda["Sirius"].souhvezdi = "Canis Major";
hvezda["Rigil Kentaurus"].souhvezdi = "Centaurus";
hvezda["Deneb"].souhvezdi = "Cygnus";
hvezda["Vega"].souhvezdi = "Lyra";
hvezda["Altair"].souhvezdi = "Aquila";

hvezda["Polaris"].typ = "Double/Cepheid";
hvezda["Mizar"].typ = "Spectroscopic Binary";
hvezda["Aldebaran"].typ = "Irregular Variable";
hvezda["Rigel"].typ = "Supergiant with Companion";
hvezda["Castor"].typ = "Multiple/Spectroscopic";
hvezda["Albireo"].typ = "Double";
hvezda["Acrux"].typ = "Double";
hvezda["Gemma"].typ = "Eclipsing Binary";
hvezda["Procyon"].typ = "Double";
hvezda["Sirius"].typ = "Double";
hvezda["Rigil Kentaurus"].typ = "Double";
hvezda["Deneb"].typ = "Supergiant";
hvezda["Vega"].typ = "White Dwarf";
hvezda["Altair"].typ = "White Dwarf";

hvezda["Polaris"].spektralniTyp = "F7";
hvezda["Mizar"].spektralniTyp = "A1 V";
hvezda["Aldebaran"].spektralniTyp = "K5 III";
hvezda["Rigel"].spektralniTyp = "B8 Ia";
hvezda["Castor"].spektralniTyp = "A1 V";
hvezda["Albireo"].spektralniTyp = "K3 II";
hvezda["Acrux"].spektralniTyp = "B1 IV";
hvezda["Gemma"].spektralniTyp = "A0 V";
hvezda["Procyon"].spektralniTyp = "F5 IV";
hvezda["Sirius"].spektralniTyp = "A1 V";
hvezda["Rigil Kentaurus"].spektralniTyp = "G2 V";
hvezda["Deneb"].spektralniTyp = "A2 Ia";
hvezda["Vega"].spektralniTyp = "A0 V";
hvezda["Altair"].spektralniTyp = "A7 V";

hvezda["Polaris"].velikost = 2.0;
hvezda["Mizar"].velikost = 2.3;
hvezda["Aldebaran"].velikost = 0.85;
hvezda["Rigel"].velikost = 0.12;
hvezda["Castor"].velikost = 1.58;
hvezda["Albireo"].velikost = 3.1;
hvezda["Acrux"].velikost = 0.8;
hvezda["Gemma"].velikost = 2.23;
hvezda["Procyon"].velikost = 0.38;
hvezda["Sirius"].velikost = -1.46;
```

```
hvezda["Rigil Kentaurus"].velikost = -0.01;
hvezda["Deneb"].velikost = 1.25;
hvezda["Vega"].velikost = 0.03;
hvezda["Altair"].velikost = 0.77;
```

Jak můžete vidět, opakuje se ve výpisu 8.1 mnoho podobného programového kódu. Každá hvězda se definuje a následně se jí přiřadí čtyři atributy – souhvězdí, ve kterém se nachází, její typ, její spektrální typ a její velikost.

Nyní se podívejte na programový kód ve výpisu 8.2. Plní stejnou úlohu jako kód z výpisu 8.1, ale tentokrát s využitím třídy.

Výpis 8.2. Vytvoření objektu hvězdy pomocí třídy `Hvezda`

```
var hvezda = {};

function Hvezda(souhvezdi,typ,spektralniTyp,velikost) {
    this.souhvezdi = souhvezdi;
    this.typ = typ;
    this.spektralniTyp = spektralniTyp;
    this.velikost = velikost;
}

hvezda["Polaris"] = new Hvezda("Ursa Minor","Double/Cepheid","F7",2.0);
hvezda["Mizar"] = new Hvezda("Ursa Major","Spectroscopic Binary",
  "A1 V",2.3);
hvezda["Aldebaran"] = new Hvezda("Taurus","Irregular Variable",
  "K5 III",0.85);
hvezda["Rigel"] = new Hvezda("Orion","Supergiant with Companion",
  "B8 Ia",0.12);
hvezda["Castor"] = new Hvezda("Gemini","Multiple/Spectroscopic",
  "A1 V",1.58);
hvezda["Albireo"] = new Hvezda("Cygnus","Double","K3 II",3.1);
hvezda["Acrux"] = new Hvezda("Crux","Double","B1 IV",0.8);
hvezda["Gemma"] = new Hvezda("Corona Borealis","Eclipsing Binary",
  "A0 V",2.23);
hvezda["Procyon"] = new Hvezda("Canis Minor","Double","F5 IV",0.38);
hvezda["Sirius"] = new Hvezda("Canis Major","Double","A1 V",-1.46);
hvezda["Rigil Kentaurus"] = new Hvezda("Centaurus","Double","G2 V",-0.01);
hvezda["Deneb"] = new Hvezda("Cygnus","Supergiant","A2 Ia",1.25);
hvezda["Vega"] = new Hvezda("Lyra","White Dwarf","A0 V",0.03);
hvezda["Altair"] = new Hvezda("Aquila","White Dwarf","A7 V",0.77);
```

Zde uvedená funkce, ve výpisu 8.2 vyznačená tučně, vytvoří třídu hvězd s názvem `Hvezda`. Po jejím zavolání je výsledkem nový objekt `Hvezda`:

```
hvezda["Polaris"] = new Hvezda("Ursa Minor","Double/Cepheid","F7",2.0);
```

Jak vidíte, ačkoli programové kódy z obou výpisu jsou z funkčního hlediska ekvivalentní, je kód z výpisu 8.2 mnohem kratší a snadněji pochopitelný. Jen si představte objekt, který má devět atributů namísto pouze čtyř.

Vytváření objektů

Objekt se vytváří dvěma způsoby:

- Pomocí klíčového slova new, následovně:

  ```
  var hvezda = new Object;
  ```

- Nebo s využitím složených závorek, následovně:

```
var hvezda = {};
```

Který z těchto způsobů použijete záleží především na vaší osobní preferenci, protože oba slouží k témuž.

Přidání atributů k objektu

Poté, co je objekt vytvořený, je možné mu začít přiřazovat atributy a definovat metody, které nabízí. Pokud máte pouze jeden objekt, např. `hvezda`, můžete mu přiřadit atributy přímo následujícím způsobem:

```
hvezda.nazev = "Polaris";
hvezda.souhvezdi = "Ursa Minor";
```

Předchozí část kapitoly ukázala, jak vytvořit více objektů s několika atributy a jak použít třídu pro efektivní vytváření objektů.

Zobrazení atributů objektu

Pomocí cyklu `for...in` je možné projít všechny atributy objektu. Vyzkoušejte to:

Průchod před atributy objektu

1. Pomocí editoru Microsoft Visual Studio, Eclipse nebo jiného editoru upravte soubor *pruchodAtributy.htm* nacházející se v adresáři *Kapitola08* s ukázkovými kódy.
2. Do stránky vložte následující programový kód, znázorněný tučně:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Atributy objektu</title>

<script type = "text/javascript">
  var hvezda = {};

  function Hvezda(souhvezdi,typ,spektralniTyp,velikost) {
      this.souhvezdi = souhvezdi;
      this.typ = typ;
      this.spektralniTyp = spektralniTyp;
      this.velikost = velikost;
  }

  hvezda["Polaris"] = new Hvezda("Ursa Minor",
    "Double/Cepheid","F7",2.0);
  hvezda["Mizar"] = new Hvezda("Ursa Major","Spectroscopic Binary","A1 V",2.3);
  hvezda["Aldebaran"] = new Hvezda("Taurus","Irregular Variable","K5 III",0.85);
  hvezda["Rigel"] = new Hvezda("Orion","Supergiant with Companion","B8 Ia",0.12);
  hvezda["Castor"] = new Hvezda("Gemini","Multiple/Spectroscopic","A1 V",1.58);
  hvezda["Albireo"] = new Hvezda("Cygnus","Double","K3 II",3.1);
  hvezda["Acrux"] = new Hvezda("Crux","Double","B1 IV",0.8);
  hvezda["Gemma"] = new Hvezda("Corona Borealis","Eclipsing Binary","A0 V",2.23);
  hvezda["Procyon"] = new Hvezda("Canis Minor","Double","F5 IV",0.38);
```

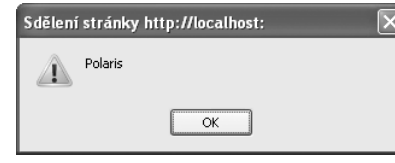
```
    hvezda["Sirius"] = new Hvezda("Canis Major","Double","A1 V",-1.46);
    hvezda["Rigil Kentaurus"] = new Hvezda("Centaurus","Double","G2 V",-0.01);
    hvezda["Deneb"] = new Hvezda("Cygnus","Supergiant","A2 Ia",1.25);
    hvezda["Vega"] = new Hvezda("Lyra","White Dwarf","A0 V",0.03);
    hvezda["Altair"] = new Hvezda("Aquila","White Dwarf","A7 V",0.77);

</script>
</head>
<body>
<script type = "text/javascript" >

for (var atribut in hvezda) {
    alert(atribut);
}

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

3. Otevřete stránku ve webovém prohlížeči. Postupně se zobrazí dialogové okno pro každý z atributů objektu `hvezda`, celkem 14 (ano, je to spousta klepání myší, omlouvám se za to). Jak vypadá jedno z dialogových oken, ukazuje obrázek vpravo.



Tento příklad je založený na dřívějším příkladu použití třídy pro vytváření objektů. V tomto případě se objekt `hvezda` vytváří pomocí následujícího kódu:

```
var hvezda = {};
```

Tomuto objektu se následně přiřadí několik atributů reprezentujících názvy jednotlivých hvězd, jejichž hodnotami jsou objekty `Hvezda` (vytvořené pomocí třídy):

```
hvezda["Polaris"] = new Hvezda("Ursa Minor","Double/Cepheid","F7",2.0);
```

Každý z atributů objektu `hvezda`, v tomto případě názvy jednotlivých hvězd, se poté projde pomocí cyklu `for...in` nacházejícího se v elementu `body`:

```
for (var atribut in hvezda) {
    alert(atribut);
}
```

Možná si říkáte, jak získat atributy samotných hvězd, tj. objektů `Hvezda`, kterými jsou `souhvezdi`, `typ`, `spektralniTyp` a `velikost`. Jak projít těmito atributy ukáže kapitola 11.

Hledání atributu

Někdy nechcete nebo nepotřebujete procházet všemi atributy objektu. Někdy si prostě přejete vědět, jestli daný atribut v objektu již existuje. K tomuto účelu je možné použít operátor `in`, tak jak ukazuje následující programový kód:

```
if (atribut in objekt) {
    // proveď nějakou akci
}
```

Kompletnější příklad ukazuje výpis 8.3, ve kterém se ověřuje, jestli objekt `hvezda` má za atribut jeden z názvů hvězd, `Polaris`, a pokud ano, přidá se objektu nový atribut.

Výpis 8.3. Hledání atributu

```
var hvezda = {};

function Hvezda(souhvezdi,typ,spektralniTyp,velikost) {
    this.souhvezdi = souhvezdi;
    this.typ = typ;
    this.spektralniTyp = spektralniTyp;
    this.velikost = velikost;
}

hvezda["Polaris"] = new Hvezda("Ursa Minor","Double/Cepheid","F7",2.0);
hvezda["Mizar"] = new Hvezda("Ursa Major","Spectroscopic Binary","A1 V",2.3);
hvezda["Aldebaran"] = new Hvezda("Taurus","Irregular Variable","K5 III",0.85);
hvezda["Rigel"] = new Hvezda("Orion","Supergiant with Companion","B8 Ia",0.12);
hvezda["Castor"] = new Hvezda("Gemini","Multiple/Spectroscopic","A1 V",1.58);
hvezda["Albireo"] = new Hvezda("Cygnus","Double","K3 II",3.1);
hvezda["Acrux"] = new Hvezda("Crux","Double","B1 IV",0.8);
hvezda["Gemma"] = new Hvezda("Corona Borealis","Eclipsing Binary","A0 V",2.23);
hvezda["Procyon"] = new Hvezda("Canis Minor","Double","F5 IV",0.38);
hvezda["Sirius"] = new Hvezda("Canis Major","Double","A1 V",-1.46);
hvezda["Rigil Kentaurus"] = new Hvezda("Centaurus","Double","G2 V",-0.01);
hvezda["Deneb"] = new Hvezda("Cygnus","Supergiant","A2 Ia",1.25);
hvezda["Vega"] = new Hvezda("Lyra","White Dwarf","A0 V",0.03);
hvezda["Altair"] = new Hvezda("Aquila","White Dwarf","A7 V",0.77);

if ("Polaris" in hvezda) {
    hvezda["Polaris"].aka = "Severka";
    alert("Polárka je též známá jako " + hvezda["Polaris"].aka);
}
```

Poznámka: Existují i další způsoby, jak ověřit existenci atributu objektu, kterou tato kniha nezmiňuje. K tomuto účelu lze použít např. operátor `!==`.

Přidání metod k objektu

Stejně jako se k objektu přidávaly atributy, je možné k němu přidávat také metody. Např. třídu používanou v předchozích příkladech je možné rozšířit tak, aby nabízela metodu `zobraz`, která zajistí zobrazení dialogového okna. Tuto metodu lze přirozeně rozšířit jakkoli si přejete. Podívejte se na následující programový kód:

```
function Hvezda(souhvezdi,typ,spektralniTyp,velikost) {
    this.souhvezdi = souhvezdi;
    this.typ = typ;
    this.spektralniTyp = spektralniTyp;
    this.velikost = velikost;
    this.zobraz = function zobraz() {
        alert("Došlo k zavolání metody zobraz.");
    }
}
```

Volání této metody vypadá následovně:

```
hvezda["Polaris"].zobraz();
```

Studium objektově orientovaného programování v JavaScriptu tímto není u konce. Více pokročilejší funkce objektově orientovaného programování, jako je dědičnost nebo prototypy, jsou v JavaScriptu také k dispozici, ale spadají již za rámec této knihy. MSDN magazín publikoval článek o některých pokročilejších konceptech, který je možné nalézt na adrese *http://msdn.microsoft.com/msdnmag/issues/07/05/JavaScript/default.aspx*.

Více o polích

Jak probrala kapitola 4, Práce s proměnnými a daty, umožňují pole seskupit skupinu hodnot do objektu a následně k jednotlivým hodnotám přistupovat pomocí číselného indexu. Kapitola 4 ukázala několik způsobů definice polí. Jedním z nich je explicitní použití konstruktoru `Array`, následovně:

```
var hvezda = new Array( );
hvezda[0] = "Polaris";
hvezda[1] = "Deneb";
hvezda[2] = "Vega";
hvezda[3] = "Altair";
```

Totéž lze provést také pomocí implicitního konstruktoru pole (hranatých závorek), takto:

```
var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];
```

Atribut length

Atribut `length` objektu pole obsahuje počet prvků pole. Existuje významný rozdíl mezi tím, kolik prvků pole obsahuje a kolik jich bylo definováno. Zde je jednoduchý příklad. Vyjdeme z definice pole `hvezda` uvedené výše. Je možné napočítat čtyři prvky: Polaris, Deneb, Vega a Altair. Atribut `length` tento počet potvrzuje:

```
var pocetHvezd = hvezda.length;   // pocetHvezd = 4
```

Poznámka: Pomocí atributu `length` je možné spočítat i prvky pole, které doposud nebyly definovány ani inicializovány.

Metody objektu pole

Abyste se seznámili s metodami poskytovanými objektem pole, zaměřuje se tato část kapitoly na některé z těchto metod. Více informací naleznete ve specifikaci ECMA-262 dostupné na adrese *http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf*.

Přidávání a odebírání prvků pole

Prvky je do pole možné přidávat jak na začátek, tak na konec pole, pomocí několika metod.

Použití metody concat pro přidání prvků

Metoda `concat` přidává prvek na konec pole. Metoda bere jako parametr přidávaný prvek a vrací nové pole, takto:

```
var mePole = new Array( );
mePole[0] = "prvni";
mePole[1] = "druhy";
var novePole = mePole.concat("treti");
// pole novePole nyní obsahuje [prvni,druhy,treti]
```

K poli je možné připojit také jiné pole, takto:

```
var mePrvniPole = [51,67];
var meDruhePole = [18,"ahoj",125];
var novePole = mePrvniPole.concat(meDruhePole)
// pole novePole nyní obsahuje [51,67,18,"ahoj",125]
```

Přidávání prvků do pole pomocí metody `concat`

1. Pomocí editoru Microsoft Visual Studio, Eclipse nebo jiného editoru upravte soubor *concat.htm* nacházející se v adresáři *Kapitola08* s ukázkovými kódy.
2. Do stránky vložte následující programový kód, znázorněný tučně:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Metoda concat</title>
    <script type = "text/javascript">

    var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];

    for (var i = 0; i < hvezda.length; i++) {
        alert(hvezda[i]);
    }

    </script>
</head>
<body>
<p>Příklad z kapitoly 8</p>
</body>
</html>
```

3. Otevřete stránku ve webovém prohlížeči. Pro každou ze čtyř hvězd uvedených v poli `hvezda` se zobrazí dialogové okno (jako to zobrazené na obrázku vpravo).
4. Nyní přidáme do pole `hvezda` další hvězdy (jistě, je možné je přidat přímo, ale o to tu nejde). Zde je programový kód (změny jsou vyznačeny tučně):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Metoda concat</title>
    <script type = "text/javascript">

    var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];

    var noveHvezdy = ["Aldebaran", "Rigel"];
    var viceHvezd = hvezda.concat(noveHvezdy);

    for (var i = 0; i < viceHvezd.length; i++) {
        alert(viceHvezd[i]);
    }

    </script>
```

```
</head>
<body>
<p>Příklad z kapitoly 8</p>
</body>
</html>
```

5. Uložte stránku a otevřete ji ve webovém prohlížeči. Nyní se postupně zobrazí šest dialogových oken (pardon), jedno pro každou z uvedených hvězd, jako je např. toto pro hvězdu Aldebaran.

Přidávání prvků do pole pomocí metody `join`

Metoda `join` převede všechny prvky pole do jednoho řetězce. Tato metoda není jako metoda `concat`, která provádí spojení polí a žádnou konverzi. Zde je ukázkový programový kód:

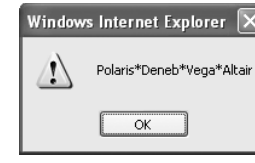
```
var hvezda = ["Polaris","Deneb","Vega","Altair"];
var hvezdaRetezec = hvezda.join();
alert(hvezdaRetezec);
```

Obrázek 8.1. Použití metody `join` pro spojení prvků pole do řetězce

Proměnná `hvezdaRetezec` bude mít hodnotu `Polaris,Deneb,Vega,Altair`, jak také ukazuje obrázek 8.1.

Metoda `join` dovoluje také specifikovat oddělovač jednotlivých prvků pole. Namísto použití výchozí čárky si můžete např. přát použít hvězdičku, takto:

```
var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];
var hvezdaRetezec = hvezda.join("*");
alert(hvezdaRetezec);
```



Obrázek 8.2. Spojení prvků pole do řetězce s využitím vlastního oddělovače

Tip: Metoda `join` představuje rychlý způsob, jak zobrazit celý obsah pole, aniž by bylo nutné vytvářet cyklus `for`.

Použití metod push a pop pro přidání, resp. odebrání, prvků pole

Zatímco metoda `concat` vrací nově vytvořené pole, metody `push` a `pop` sloužící pro přidání, resp. odebrání, prvku z pole vrací novou délku pole, resp. odebraný prvek. Metody `push` a `pop` operují na konci pole:

```
var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];
hvezda.push("Aldebaran");
```

Po provedení tohoto programového kódu bude pole `hvezda` obsahovat prvky Polaris, Deneb, Vega, Altair, Aldebaran.

Metoda `pop` odstraní z pole poslední prvek a vrátí odstraněný prvek:

```
var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];
var odstranenyPrvek = hvezda.pop();
```

Proměnná `odstranenyPrvek` by obsahovala řetězec `Altair`, protože se jednalo o poslední prvek pole. Délka pole by se také zmenšila o 1.

Použití metod shift a unshift pro přidání, resp. odebrání, prvků pole

Metody `push` a `pop` operují na konci pole. Metody `unshift` a `shift` provádí totéž jako metody `push` a `pop`, pouze na opačném konci pole. Metoda `unshift` přidá prvek na začátek pole:

```
var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];
hvezda.unshift("Aldebaran");
```

Pole `hvezda` by po provedení tohoto kódu vypadalo následovně:

```
["Aldebaran", "Polaris", "Deneb", "Vega", "Altair"]
```

Nyní použijeme metodu `shift` pro odstranění prvku ze začátku pole:

```
var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];
var odstranenyPrvek = hvezda.shift();
```

Pole `hvezda` by po provedení tohoto kódu vypadalo následovně:

```
["Deneb", "Vega", "Altair"]
```

Použití metody slice pro získání části pole

Metoda `slice` je užitečná pokud potřebujete získat specifickou část (výřez) pole. Např. následující programový kód uloží do proměnné `vyrezHvezda` pole s prvky `Vega` a `Altair`, protože se jedná o v pořadí třetí, resp. čtvrtý, prvek pole `hvezda` (pamatujte, že se prvky pole indexují od nuly). Programový kód je následující:

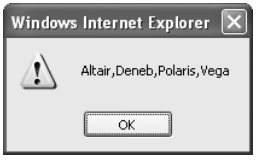
```
var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];
var vyrezHvezda = hvezda.slice(2);
```

Třídění prvků pomocí metody sort

Někdy se může hodit setřídit prvky pole. Podívejte se na následující programový kód:

```
var hvezda = ["Polaris", "Deneb", "Vega", "Altair"];
var setrideno = hvezda.sort();
```

Výsledek ukazuje obrázek 8.3, a jak můžete vidět, jsou prvky pole `hvezda` nyní setříděné podle abecedy, ačkoli v programovém kódu podle abecedy zadány nejsou. Všimněte si, že jak původní pole `hvezda`, tak pole uložené v proměnné `setrideno`, jsou setříděná.



Obrázek 8.3. Výsledek setřídění pole metodou `sort`

Metodu `sort` nepoužívejte pro setřídění čísel. Podívejte se na následující programový kód:

```
var cisla = [11,543,22,111];
var setrideno = cisla.sort();
```

Logické by bylo, kdyby pole setrideno obsahovalo popořadě prvky 11,22,111 a 543, ale namísto toho dojde k setřídění pole podle abecedy, tak jak ukazuje obrázek 8.4.

Obrázek 8.4. Pokus o setřídění číselných prvků pole pomocí metody `sort` neuspěje – alespoň ne pokud je chcete setřídit podle číselné hodnoty

Objekt pole nabízí ještě další metody, o kterých byste měli vědět, ale často je nejspíš používat nebudete (záleží na požadavcích vaší webové stránky). Pro více informací prostudujte standard ECMA-262. Dvě z těchto metod, se kterými se můžete setkat, naleznete v tabulce 8.1.

Tabulka 8.1. Vybrané metody objektu pole (Array)

Metoda	Popis
`reverse`	Obrátí pořadí prvků v poli.
`splice`	Vloží nebo odstraní prvky z pole.

Vestavěné objekty

Jazyk JavaScript nabízí několik užitečných objektů, které programům v JavaScriptu pomáhají při běžných úkolech. Mezi těmito objekty jsou některé, které jsme již diskutovali dříve, jako např. objekty `Number` nebo `Math` probírané v kapitole 4.

Tato část kapitoly se zaměřuje na objekt `Date`. Ve zbytku knihy si představíme také objekty `Regexp` a `String`, až to bude zapotřebí.

Objekt Date

Objekt `Date` nabízí řadu metod užitečných při práci s daty v JavaScriptu. Ve skutečnosti je těchto metod až příliš na to, aby mělo v knize pro začátečníky jako je tato smysl je probírat. Proto si ukážeme některé příklady použití, jež si pravděpodobně budete přát zakomponovat do svých projektů.

Obrázek 8.5. Objekt `Date` vrací řetězec s aktuálním datem, který je možné lokalizovat prostřednictvím jeho metody `toLocaleDateString`

Zde je jednoduchý programový kód pro získání aktuálního data upraveného podle lokální časové zóny a automaticky naformátovaného pomocí metody `toLocaleDateString`:

```
var meDatum = new Date();
alert(meDatum.toLocaleDateString());
```

Po provedení tohoto kódu se zobrazí datum, podobně jako na obrázku 8.5.

> **Jaké je datum?**
>
> Všimněte si data uvedeného na obrázku 8.5, které je 16. června 2001. Můžete se ptát, jestli jsem knihu napsal v roce 2001. Ne, tak tomu není. Tento obrázek jen ilustruje problém s určováním dat v JavaScriptu. Data získaná funkcemi JavaScriptu zcela závisí na počítači, na kterém se programový kód v JavaScriptu provádí.
>
> Stalo se tedy to, že jsem si změnit datum na svém počítači na 16. června 2001, abych tento problém ilustroval. Kdykoli proto použijete metody objektu `Date`, mějte na paměti, že zcela závisí na datu nastaveném na hostitelském počítači (shodou náhod je 16. června 2001 datum mé svatby, takže nyní mám dobrou referenci, pro případ, že bych toto datum zapomenul – samozřejmě ne že by k tomu někdy mohlo dojít).

Konstruktoru objektu `Date` je možné předat řadu parametrů, od žádného až po sedm. Když se konstruktoru objektu `Date` předá jediný parametr typu řetězec, předpokládá se, že jde o řetězec obsahující datum. Pokud se mu předá parametr typ číslo, předpokládá se, že se jedná o datum uvedené v milisekundách uplynulých od 1.1.1970. V případě, že se konstruktoru předá všech sedm parametrů, mají následující význam:

```
new Date(rok, měsíc, den, hodina, minuta, sekunda, milisekunda);
```

Poznámka: Povinné jsou pouze parametry `rok` a `měsíc`, ostatní jsou volitelné.

Při práci s objektem `Date` mějte na paměti následující:

- Rok by se měl zadávat jako čtyřmístné číslo, ledaže si přejete specifikovat rok v letech 1900 až 1999. V takovém případě stačí zadat dvoumístné číslo, tzn. 0 až 99, které se přičte k 1900. Tzn. pro zadání roku 2008 je třeba uvést 2008, ale pro rok 1998 stačí uvést jen 98.

- Měsíc se reprezentuje celým číslem v rozsahu 0 až 11, kde 0 je leden a 11 prosinec.
- Den se reprezentuje celým číslem v rozsahu 1 až 31.
- Hodiny se reprezentují celým číslem v rozsahu 0 až 23, kde 23 představuje 11 hodin večer.
- Minuty i sekundy se reprezentují celým číslem v rozsahu 0 až 59.
- Milisekundy se reprezentují celým číslem v rozsahu 0 až 999.

Následující příklad sice používá některé prvky, jež vysvětlí až pozdější kapitoly, ale protože nyní probíráme objekt `Date`, je jistě vhodné ukázat si, jak vypsat aktuální datum a čas ve webové stránce. Jedná se o vcelku populární a často používanou funkci. Toto cvičení ukáže celý postup, včetně prvků, které nevysvětlím hned, ale až později.

Vypsání data a času do webové stránky

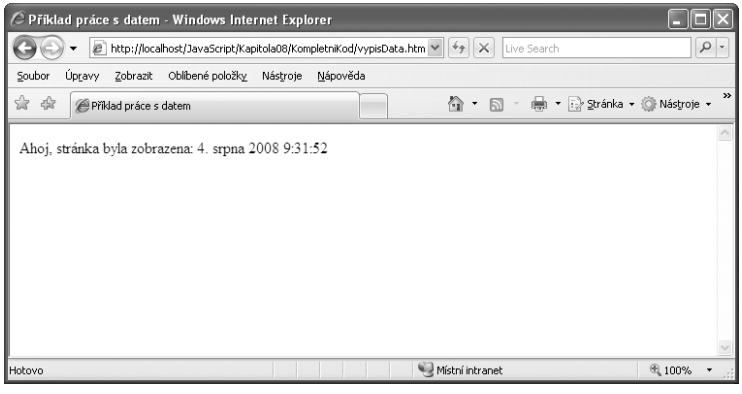
1. Pomocí editoru Microsoft Visual Studio, Eclipse nebo jiného editoru upravte soubor *vypisData.htm* nacházející se v adresáři *Kapitola08* s ukázkovými kódy.
2. Do stránky vložte následující programový kód, znázorněný tučně:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
     <title>Příklad práce s datem</title>
</head>
<body>
     <p id="vypisDatum"> </p>
     <script type = "text/javascript">

     var meDatum = new Date();
     var retezecData = meDatum.toLocaleDateString() + " " +
       meDatum.toLocaleTimeString();
     var umisteniData = document.getElementById("vypisDatum");
     umisteniData.innerHTML = "Ahoj, stránka byla zobrazena: " + retezecData;
     </script>

</body>
</html>
```

3. Když stránku uložíte a otevřete ve webovém prohlížeči, zobrazí se stránka podobná té znázorněné na následujícím obrázku (uvedené datum a čas budou samozřejmě odlišné).



Relevantní programový kód v JavaScriptu z tohoto cvičení si ještě jednou zopakujme zde:

```
var meDatum = new Date();
var retezecData = meDatum.toLocaleDateString() + " " +
  meDatum.toLocaleTimeString();
var umisteniData = document.getElementById("vypisDatum");
umisteniData.innerHTML = "Ahoj, stránka byla zobrazena: " + retezecData;
```

Programový kód spojený s objektem `Date` je vcelku jednoduchý. Používá metodu `toLocaleDateString`, kterou již znáte, a její příbuznou `toLocaleTimeString`, jež vrací lokální čas. Návratové hodnoty těchto dvou metod se spojí do jednoho řetězce, s využitím mezery jako oddělovače, a výsledek se uloží do proměnné `retezecData`, takto:

```
var retezecData = meDatum.toLocaleDateString() + " " +
  meDatum.toLocaleTimeString();
```

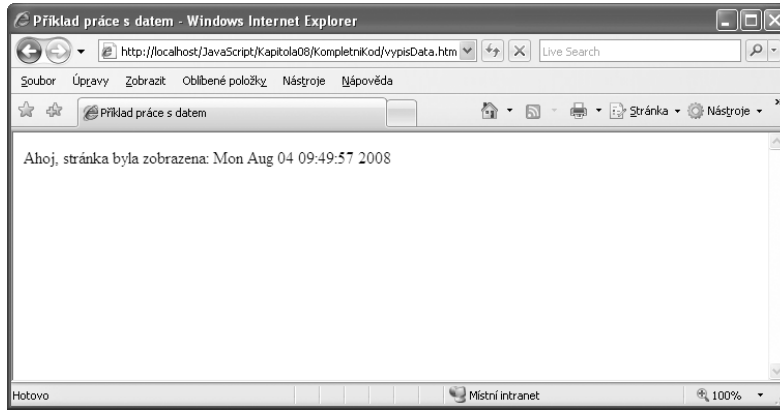
Zbytek programového kódu vypíše obsah proměnné `retezecData` do webové stránky. Více informací o tomto aspektu JavaScriptu najdete v kapitole 10, *Objektový model dokumentu*.

Další postupy pro výpis data

Postup znázorněný v předchozím příkladu není jedinou možností, jak ve stránce vypsat datum. Existují další možnosti, jejichž výběr závisí čistě na preferenci vývojáře (jinými slovy neexistuje žádný postup, který by bylo možné označit za nejlepší). Jednou z možností je zavolat `Date` jako funkci namísto vytvoření nového objektu `Date`, jak tomu bylo v předchozím příkladu. Klíčový rozdíl spočívá v nepoužití klíčového slova `new`. V případě použití tohoto klíčového slova, tzn. `new Date()`, se vytvoří nový objekt `Date`, zatímco bez tohoto klíčového slova se jedná o běžné volání funkce `Date`. Zde je odpovídající programový kód, ve kterém se `Date` volá jako funkce:

```
var meDatum = Date();
var umisteniData = document.getElementById("vypisDatum");
umisteniData.innerHTML = "Ahoj, stránka byla zobrazena: " + meDatum;
```

Pokud otevřete stránku s tímto kódem v prohlížeči, jistě si všimnete rozdílného výstupu, jak dokladuje následující obrázek:



Všimněte si, že je na obrázku nyní čas uveden v jiném, nelokalizovaném, formátu, protože nedošlo k volání metod `toLocaleDateString`, resp. `toLocaleTimeString`. Tyto rozdíly ve formátu mohou být důležité, pokud výsledek nějakým způsobem zpracováváte a ne ho pouze zobrazujete.

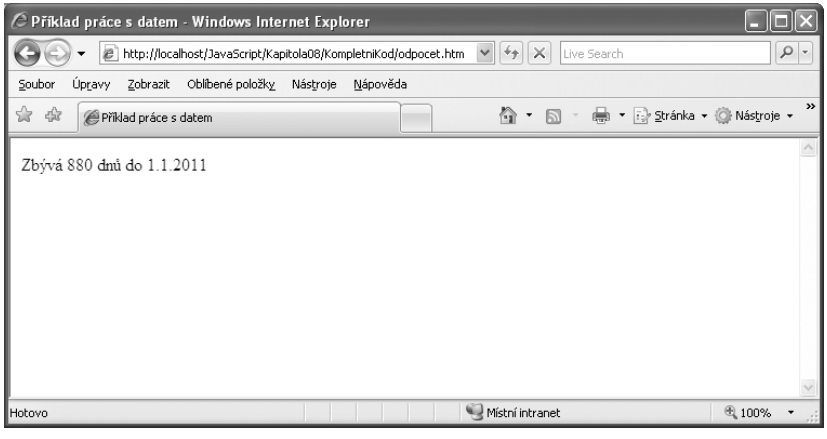
Odpočet k určitému datu v budoucnosti

1. Pomocí editoru Microsoft Visual Studio, Eclipse nebo jiného editoru upravte soubor *odpocet.htm* nacházející se v adresáři *Kapitola08* s ukázkovými kódy.
2. Do stránky vložte následující programový kód, znázorněný tučně:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Příklad práce s datem</title>
</head>
<body>
    <p id="vypisDatum"> </p>
    <script type = "text/javascript">
    var dnes = new Date();
    var potom = new Date();
    // 1.ledna 2011
    potom.setFullYear(2011,0,1);
    var rozdil = potom.getTime() - dnes.getTime();
    rozdil = Math.floor(rozdil / (1000 * 60 * 60 * 24));
    var umisteniData = document.getElementById("vypisDatum");
    umisteniData.innerHTML = "Zbývá " + rozdil + " dnů do 1.1.2011";
    </script>

</body>
</html>
```

3. Uložte stránku a otevřete ji ve webovém prohlížeči. Zobrazený počet dnů bude rozdílný, v závislosti na aktuálním datu nastaveném ve vašem počítači, ale obecně bude stránka vypadat tak, jak ukazuje následující obrázek:



Tip: Buďte opatrní, pokud v JavaScriptu používáte data na cokoli jiného než na pouhý výpis na stránce. Protože je datum zcela závislé na lokálním čase návštěvníka, neměli byste na něj spoléhat v případě čehokoli důležitého, jako je např. při vytváření objednávky.

Příklad, který jsme právě dokončili, používal některé metody jak objektu `Date`, tak objektu `Math`, jmenovitě `floor` a `getTime`. Ačkoli toho tato kniha probírá mnoho, nejedná se o kompletní referenci jazyka JavaScript. Pro tyto a další informace prostudujte MSDN, především pak článek dostupný na adrese *http://msdn2.microsoft.com/en-us/office/aa905433.aspx*, který obsahuje mnoho užitečných odkazů na další zdroje zabývající se JavaScriptem.

Poslední příklad ukazuje, jak vypočítat (přesněji řečeno hrubě odhadnout) čas potřebný pro načtení webové stránky v prohlížeči uživatele.

Poznámka: Nejedná se o zcela přesnou proceduru, protože nemůže vzít v potaz čas, který je nutný k úplnému načtení a zobrazení obrázků (nebo jiného multimediálního obsahu), jež jsou z pohledu textu webové stránky externí. Tyto jsou technicky vzato dodatečné části stránky, které se načtou poté, co se dokončí načítání stránky. Jedná se však o další běžnou funkci, se kterou jsem se setkal na mnoha webových stránkách a může se vám hodit. Tak či onak příklad ukazuje další způsob práce s daty.

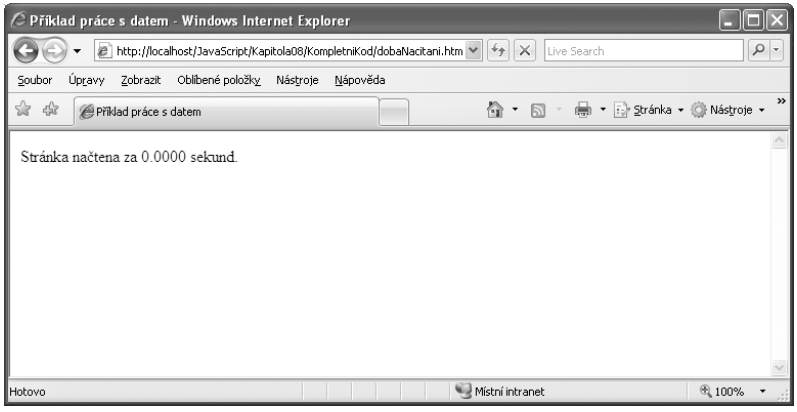
Kalkulace doby potřebné pro načtení stránky

1. Pomocí editoru Microsoft Visual Studio, Eclipse nebo jiného editoru upravte soubor *dobaNacitani.htm* nacházející se v adresáři *Kapitola08* s ukázkovými kódy.
2. Do stránky vložte následující programový kód, znázorněný tučně:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Příklad práce s datem</title>
    <script type = "text/javascript">
    var zacatek = new Date();
    var ted = zacatek.getTime();
    </script>
</head>
<body>
    <p id="vypisDatum"> </p>
    <script type = "text/javascript">
    var konec = new Date();
    var rozdil = (konec.getTime() - ted)/1000;
    var casNacteni = rozdil.toPrecision(5);
    var umisteniData = document.getElementById("vypisDatum");
    umisteniData.innerHTML = "Stránka načtena za " + casNacteni +
      " sekund.";
    </script>

</body>
</html>
```

3. Uložte stránku a otevřete ji ve webovém prohlížeči. V závislosti na rychlosti vašeho počítače, webového serveru a síťového připojení můžete stejně jako já získat stránku indikující dobu načtení 0 sekund:



4. Pokud trvá načtení vaší stránky 0 sekund, stejně jako v mém případě, je možné do stránky uměle přidat zpoždění, aby bylo možné programový kód otestovat (nikdy nedoporučuji takto postupovat u skutečné stránky, protože mě nenapadá jediný důvod proč zpomalovat zobrazování stránky. Tento postup se ale může hodit pro testovací účely). Jednoduchý způsob, jak zpomalit provádění kódu v JavaScriptu je pomocí cyklu `for`:

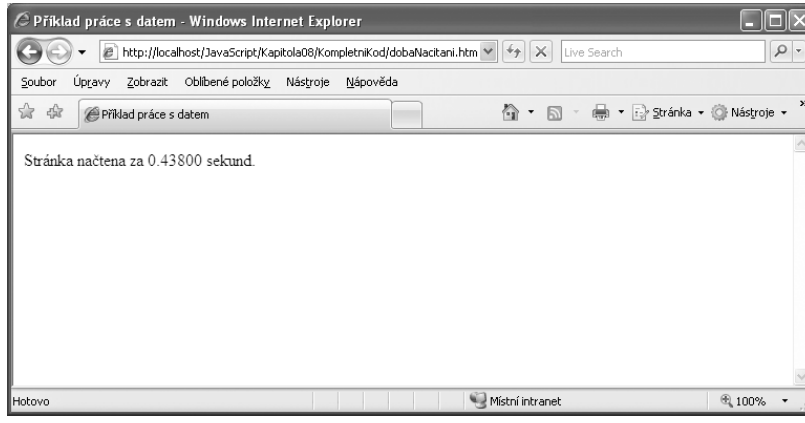
```
for (var i = 0; i < 1000000; i++) {
  // zpoždění
}
```

(Hodnota, kterou jsem zvolil, 1000000, je zcela volitelná. Můžete si přát použít větší nebo menší hodnotu, abyste docílili požadovaného zpozdění). Finální programový kód vypadá následovně:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Příklad práce s datem</title>
    <script type = "text/javascript">
    var zacatek = new Date();
    var ted = zacatek.getTime();
    for (var i = 0; i < 1000000; i++) {
        // zpoždění
    }
    </script>
</head>
<body>
    <p id="vypisDatum"> </p>
    <script type = "text/javascript">
    var konec = new Date();
    var rozdil = (konec.getTime() - ted)/1000;
    var casNacteni = rozdil.toPrecision(5);
    var umisteniData = document.getElementById("vypisDatum");
    umisteniData.innerHTML = "Stránka načtena za " + casNacteni +
      " sekund.";
    </script>

</body>
</html>
```

5. Uložte stránku a opět ji otevřete v prohlížeči. Nyní by mělo být patrné zpoždění při načítání stránky, které se projeví na zobrazené nenulové hodnotě doby načítání:



Při použití této nebo podobné funkce pro určení doby načítání stránky je důležité umístit programový kód pro určení počáteční časové hodnoty co nejblíže začátku stránky a kód pro určení koncové časové hodnoty naopak co nejblíže konci stránky, aby byl výpočet co nejpřesnější.

Nyní jste měli možnost vidět jen několik z více jak 40 metod objektu `Date`. Mnoho z těchto metod má své protějšky pro formát UTC, což znamená, že získávají, resp. nastavují, data a časy ve formátu UTC namísto lokálního formátu. Tabulka 8.2 sumarizuje metody, které vracejí data nebo čas. S výjimkou metod `getTime` a `getTimezoneOffset` mají všechny metody své protějšky pro formát UTC s názvy ve formátu `getUTCDate`, `getUTCDay` atd.

Tabulka 8.2. Metody `get` objektu `Date`

Metoda	Popis
`getDate`	Vrací den měsíce.
`getDay`	Vrací den týdne.
`getFullYear`	Vrací rok ve formátu čtyřmístného čísla.
`getHours`	Vrací hodinu.
`getMilliseconds`	Vrací milisekundu.
`getMinutes`	Vrací minutu.
`getMonth`	Vrací měsíc.
`getSeconds`	Vrací sekundu.
`getTime`	Vrací počet milisekund uplynulých od 1. ledna 1970.
`getTimezoneOffset`	Vrací počet minut určený jako rozdíl mezi formátem UTC a lokálním časem.

Mnoho `get...` metod má své příbuzné s prefixem `set`, tak jak ukazuje tabulka 8.3. A stejně jako v případě metod `get` mají i metody `set`, s výjimkou metody `setTime`, své protějšky pro formát UTC.

Tabulka 8.3. Metody `set` objektu `Date`

Metoda	Popis
`setDate`	Nastaví den měsíce.
`setFullYear`	Nastaví rok zadaný ve formátu čtyřmístného čísla. Přijímá také celá čísla reprezentující měsíc a den měsíce.
`setHours`	Nastaví hodinu.
`setMilliseconds`	Nastaví milisekundu.
`setMinutes`	Nastaví minutu.
`setMonth`	Nastaví měsíc zadaný ve formátu celého čísla.
`setSeconds`	Nastaví sekundu.
`setTime`	Nastaví čas zadaný ve formátu počtu milisekund uplynulých od 1. ledna 1970.

Mezi některé další dostupné metody patří ty, které jste již měli možnost vidět, jako `toLocaleDateString` a podobné s ní spojené jako `toLocaleString`, `toGMTString`, `toLocaleTimeString`, `toString`, `toDateString`, `toUTCString` a `toTimeString`. Vyzkoušejte si tyto metody jak budete chtít. Tyto jednoduché úryvky kódu vám pomohou do začátku. Zkuste je zadat do adresního řádku vašeho prohlížeče:

- `javascript:var meDatum = new Date(); alert(meDatum.toLocaleDateString());`
- `javascript:var meDatum = new Date(); alert(meDatum.toLocaleString());`
- `javascript:var meDatum = new Date(); alert(meDatum.toGMTString());`
- `javascript:var meDatum = new Date(); alert(meDatum.toLocaleTimeString());`
- `javascript:var meDatum = new Date(); alert(meDatum.toString());`
- `javascript:var meDatum = new Date(); alert(meDatum.toDateString());`
- `javascript:var meDatum = new Date(); alert(meDatum.toUTCString());`
- `javascript:var meDatum = new Date(); alert(meDatum.toTimeString());`

Cvičení

1. Vytvořte programový kód, který pomocí cyklu projde polem tvořeným čtyřmi prvky, ukázaným níže, a zobrazte tyto prvky pomocí funkce `alert`:
   ```
   var hvezda = ["Polaris","Deneb","Vega","Altair"];
   ```
2. Vytvořte objekt, který poslouží pro uložení informací o třech vašich oblíbených skladbách. Objekt by měl obsahovat atributy pro jméno interpreta, délku skladby a její název.
3. První příklad (krok za krokem) použil následující třídu pro vytváření objektů hvězd:
   ```
   var hvezda = {};

   function Hvezda(souhvezdi,typ,spektralniTyp,velikost) {
        this.souhvezdi = souhvezdi;
        this.typ = typ;
        this.spektralniTyp = spektralniTyp;
        this.velikost = velikost;
   }

   hvezda["Polaris"] = new Hvezda("Ursa Minor","Double/Cepheid","F7",2.0);
   hvezda["Mizar"] = new Hvezda("Ursa Major","Spectroscopic Binary",
     "A1 V",2.3);
   hvezda["Aldebaran"] = new Hvezda("Taurus","Irregular Variable",
     "K5 III",0.85);
   ```

```
hvezda["Rigel"] = new Hvezda("Orion","Supergiant with Companion",
  "B8 Ia",0.12);
hvezda["Castor"] = new Hvezda("Gemini","Multiple/Spectroscopic",
  "A1 V",1.58);
hvezda["Albireo"] = new Hvezda("Cygnus","Double","K3 II",3.1);
hvezda["Acrux"] = new Hvezda("Crux","Double","B1 IV",0.8);
hvezda["Gemma"] = new Hvezda("Corona Borealis","Eclipsing Binary",
  "A0 V",2.23);
hvezda["Procyon"] = new Hvezda("Canis Minor","Double","F5 IV",0.38);
hvezda["Sirius"] = new Hvezda("Canis Major","Double","A1 V",-1.46);
hvezda["Rigil Kentaurus"] = new Hvezda("Centaurus","Double",
  "G2 V",-0.01);
hvezda["Deneb"] = new Hvezda("Cygnus","Supergiant","A2 Ia",1.25);
hvezda["Vega"] = new Hvezda("Lyra","White Dwarf","A0 V",0.03);
hvezda["Altair"] = new Hvezda("Aquila","White Dwarf","A7 V",0.77);
```

Tento programový kód následně použil jednoduchý cyklus for pro průchod přes atributy objektu hvezda a zobrazil názvy hvězd, takto:

```
for (var atribut in hvezda) {
  alert(atribut);
}
```

Vaším úkolem je modifikovat tento programový kód tak, aby výsledkem jeho provedení bylo zobrazení jediného dialogového okna se všemi názvy hvězd namísto zobrazení dialogového okna pro každý z názvů.