
Y36WMM

Logika na straně klienta, skriptovací jazyky



Computer Graphics Group



Obsah

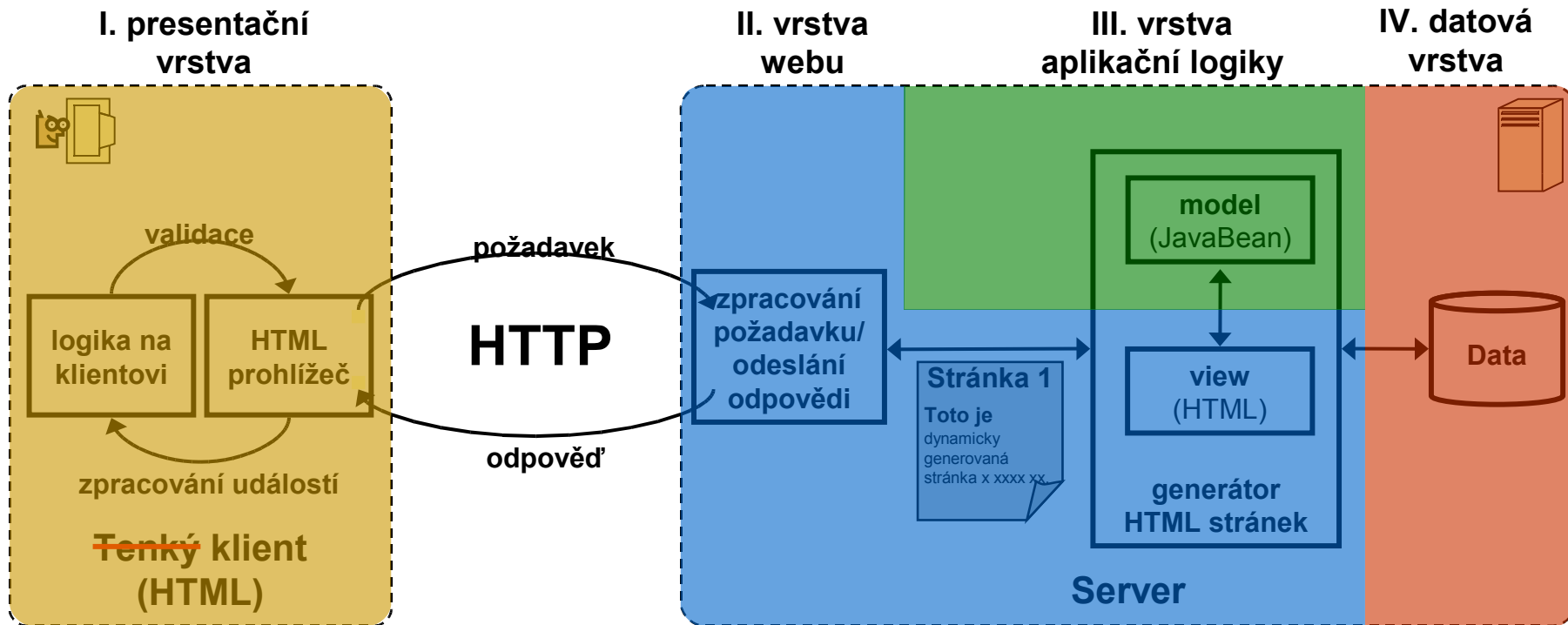
- **Architektura webapp (rekapitulace+vymezení přednášky)**
- **Co to je DHTML?**
 - stránka reaguje na události bez nutnosti spolupráce se serverovou stranou web aplikace (změna obsahu a prezentace stránky, validace formulářů, atd.)
 - směs následujících technologií: CSS, DOM, klientské skriptování
- **Co to je skriptovací jazyk?**
 - programovací jazyk (skriptování vs. programování – bible CD-32)
 - skripty pracují v předpřipraveném prostředí (datový model, UI+prezentace dat, události)
 - JavaScript – odvinut z C/C++
- **K čemu slouží a k čemu ne**
 - obecně: viz Bible str.9 a chapter 4 str. CD-8
 - kontrola vstupních dat, manipulace s malými objemy dat, dymické HTML (událost => změna HTML elementu /např. obrázků, položek ve formuláři/, generování HTML od oken)
 - příklady použití (Bible chapter 4, str. CD-3 až CD-7):
- **Vývoj DHTML**
 - poprvé: JavaScript v NN2 (MS JScript, JavaScript 1.1 -> ECMAScript 1997)
 - průlom. IE 4: umožnil manipulaci s libovolným elementem
 - CSS -> umožnily skriptům měnit prezentace již zobrazeného obsahu (změnou stylu nebo pravidla)
 - standard pro DOM: přímá manipulace skriptů s HTML obsahem

Obsah

- **DOM**
 - bible CD-8-14
- **referencování objektů v DOMu**
 - bible CD-15-18
 - name vs. ID (bible CD-15)
- **Definice elementů a manipulace s nimi**
 - properties, methods, eventHandlers (bible CD-18-21)
- **Skript v HTML**
 - element "script" (cd-24)
 - kam ho dát (cd-25-27): do head, do body, skrytí do komentářů
 - JavaScript statements (cd-27):
 - definice a inicializace proměnné, přiřazení/změna hodnoty property nebo proměnné, vyvolání metody objektu, vyvolání funkce, provedení volby
 - kdy se skript spouští (cd-28)
- **Základy programování**
 - Datové typy Javascriptu (cd-36)
 - proměnné, výrazy, konverze datových typů, operátory
 - volby a cykly (cd-48)
 - pole (cd-55), DOM v poli – forms[] (cd-59)
- **Window objekt (chap 8)**

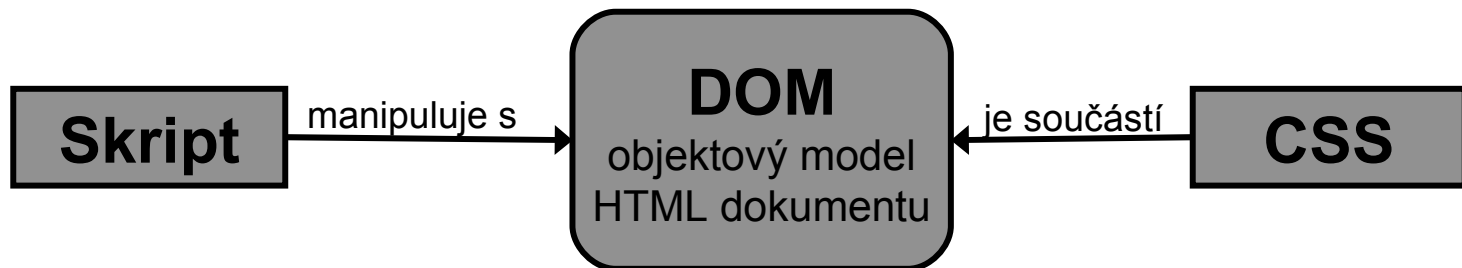


Architektura web aplikace: dynamický web



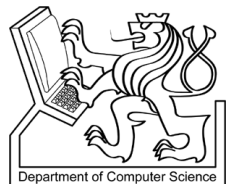
Co je to DHTML?

- **Cíl:** HTML dokument reaguje na události bez nutnosti spolupráce se serverovou stranou web aplikace
 - změna obsahu a prezentace stránky, validace formulářů, atd.
- **Řešení:** umožnit vytvářet klientský program manipulující s obsahem HTML dokumentu
- **DHTML je směs následujících technologií:**
 - DOM (Document Object Model)
 - klientské skriptování
 - CSS



Co je to skriptovací jazyk?

- **programovací jazyk**
 - JavaScript – odvinut z C/C++
- **skripty pracují v předpřipraveném prostředí**
 - datový model DOM
 - UI+prezentace dat: řeší HTML prohlížeč
 - události
- **skripty mají omezené pole působnosti**
 - bezpečnostní důvody



K čemu skripty slouží a k čemu ne?

■ ANO

- kontrola a předzpracování vstupních dat (formuláře)
- manipulace s malými objemy dat
- dynamické změny obsahu HTML
 - událost => změna HTML elementu (např. obrázků, položek ve formuláři), generování HTML do nových oken prohlížeče

■ NE

- spouštění aplikací na klientském počítači
- manipulace se soubory a adresáři

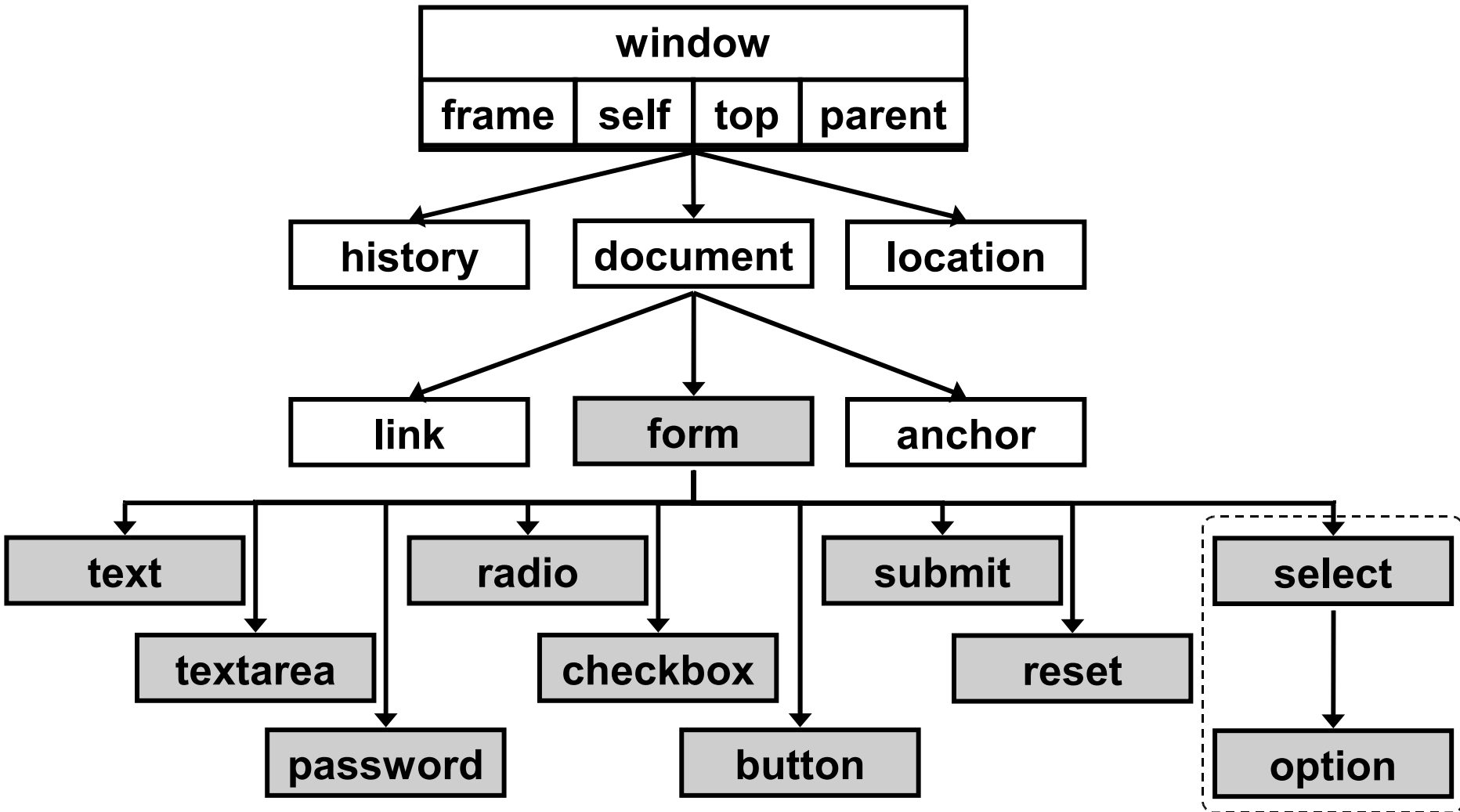
POZOR! Není-li zaručeno, že prohlížeč všech uživatelů umí spouštět skripty, vaše stránky by měly fungovat i bez nich.



Vývoj DHTML

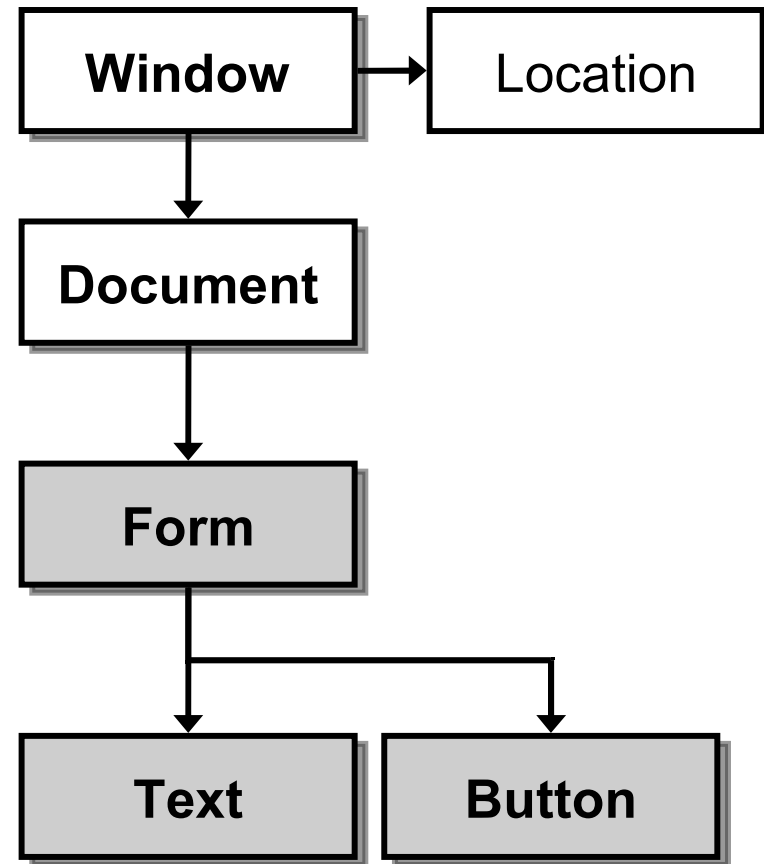
- poprvé: JavaScript v NN2
- MS JScript, JavaScript 1.1 -> ECMAScript (rok 1997)
- průlom: IE 4: umožnil manipulaci s libovolným elementem
- CSS -> umožnily skriptům měnit prezentaci již zobrazeného obsahu (změnou stylu nebo pravidla)
- definován standard pro DOM
 - umožnění přímé manipulace skriptů s HTML obsahem

DOM - hierarchie



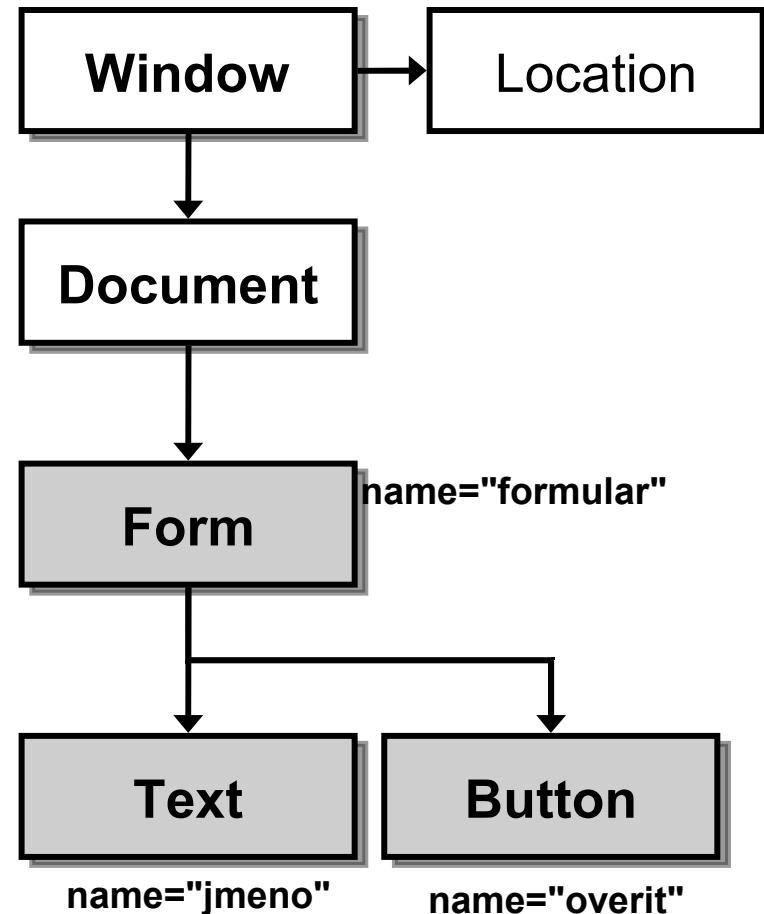
DOM: Ukázka

```
<html>
<head>
  <title>Jednoduchý
  dokument</title>
</head>
<body>
<h1>Tělo dokumentu</h1>
<form>
  <input type="text"/>
  <input type="button"/>
</form>
</body>
</html>
```

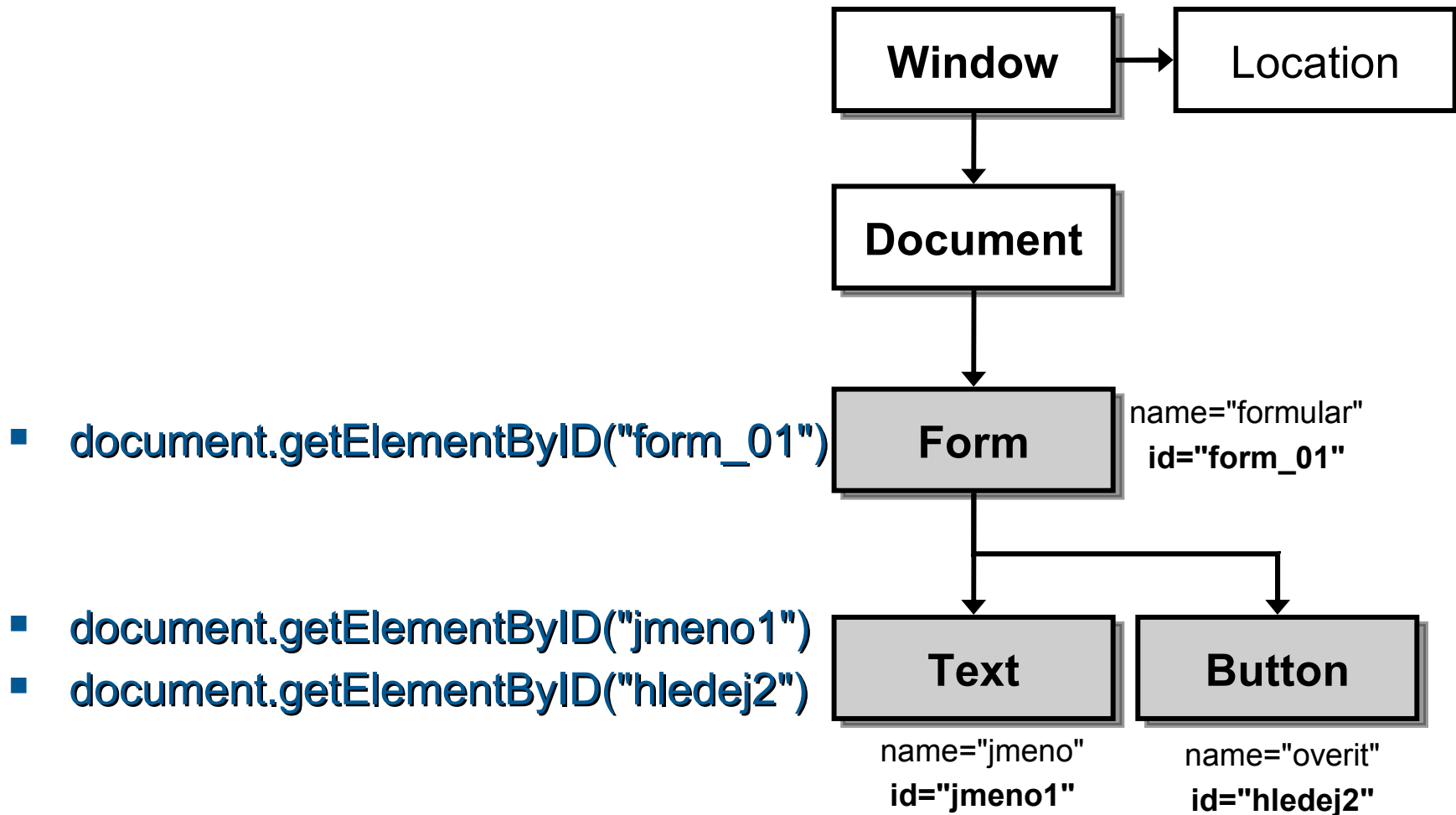


DOM: reference

- `window`
- `window.document`
- `window.document.formular`
- `window.document.forms[0]`
- `window.document.forms["formular"]`
- `window.document.formular.jmeno`
- `window.document.formular.elements[0]`
- `window.document.formular.elements["jmeno"]`
- `window.document.formular.overit`
- `window.document.formular.elements[1]`
- `window.document.formular.elements["overit"]`
- `window.document.forms[0][1]`

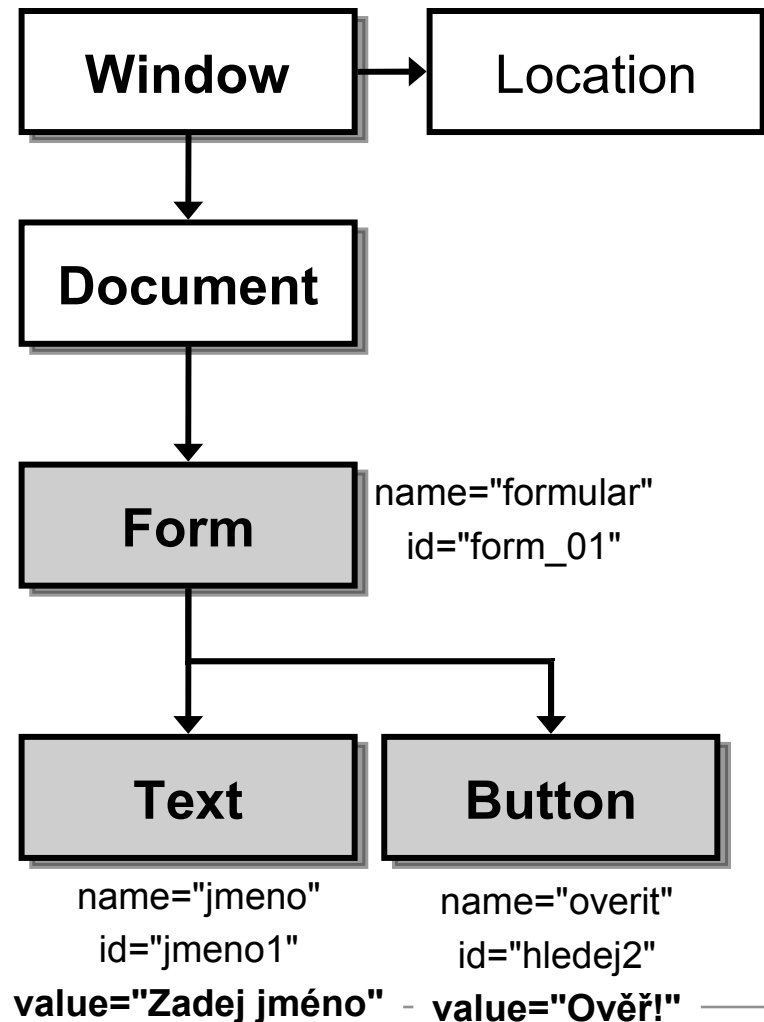


DOM: reference



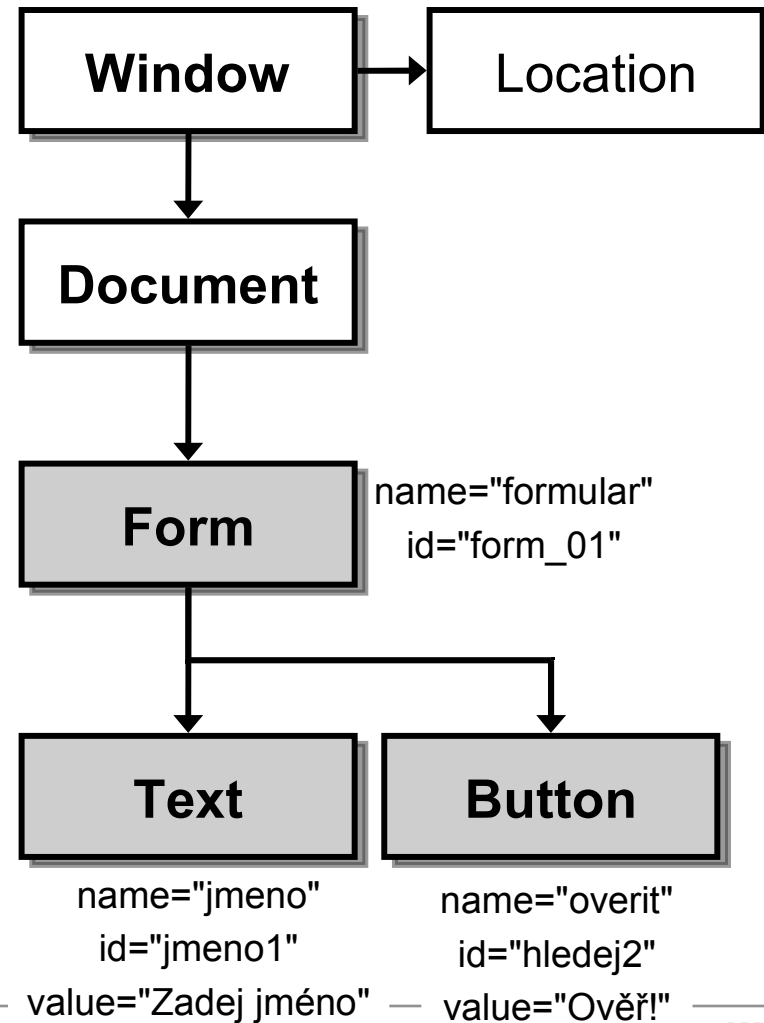
DOM: Vlastnosti (properties)

- `document.formular.id`
- `document.formular.jmeno.value`
- `document.formular.button.value`



DOM: Metody

- `window.moveTo(30,50)`
- `document.write("Nějaký text")`
- `document.formular.submit()`
- `document.formular.jmeno.select()`



DOM: Ovladače událostí

...

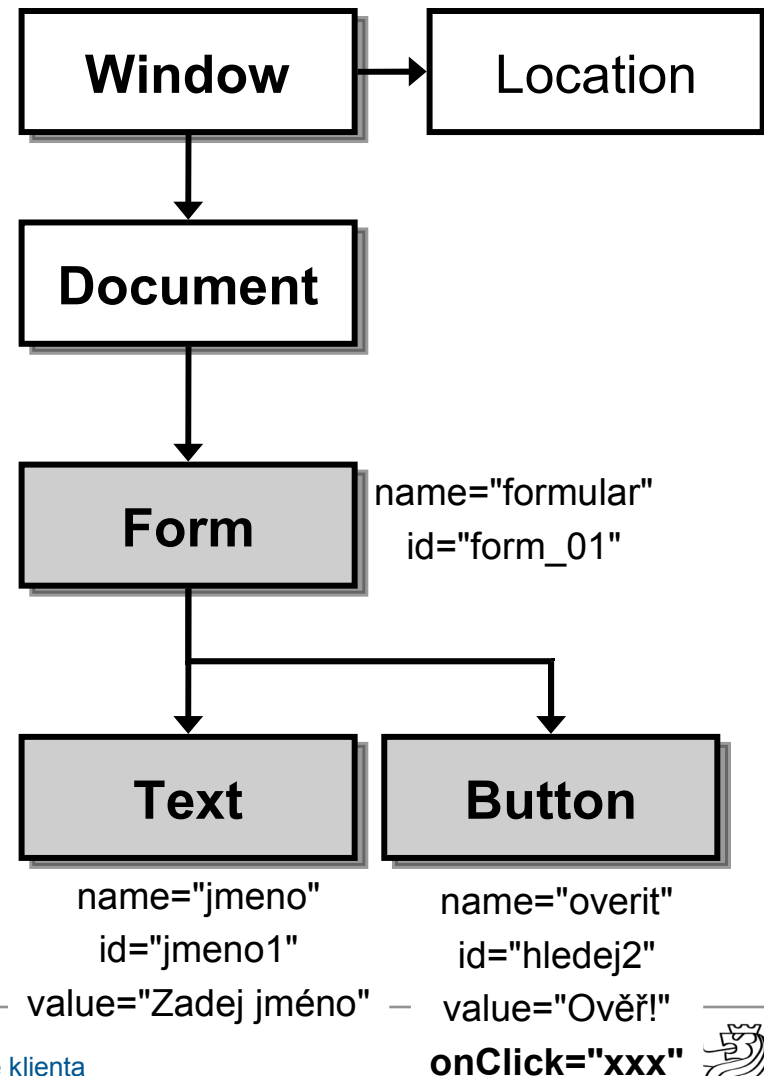
```
<form>
```

```
  <input type="text"/>
```

```
  <input type="button"  
    id="hledej2" value="Ověř!"  
    onClick="alert('Ověřuji...')" />
```

```
</form>
```

....



Skripty a HTML: jak ho zapsat

```
<script language="JavaScript" type="text/javascript">
```

```
<!--  
    tady je skript  
-->  
</script>
```

schová skript
před prohlížeči,
které ho neumí

```
<script language="JavaScript" type="text/javascript" src="skript.js"></script>
```

```
<input type="button" onClick="tady je skript"/>
```



Skripty a HTML: kam ho zapsat

```
<html>
```

```
<head>
```

```
<title>Jednoduchý dokument</title>
```

```
<script type="text/javascript">tady je skript</script>
```

```
</head>
```

reakce
na
události

```
<body>
```

```
<h1>Tělo dokumentu</h1>
```

```
<script type="text/javascript">tady je skript</script>
```

```
<form>
```

```
<input type="text"/>
```

```
<input type="button" onClick="tady je skript"/>
```

```
</form>
```

```
</body>
```

reakce na události

tvorba obsahu při
načítání

```
</html>
```



Kdy se skripty spouští

- při načítání dokumentu: uvnitř *body*

`<body>`

`<h1>Tělo dokumentu</h1>`

`<script type="text/javascript">tady je skript</script>`

`</body>`

- okamžitě po načtení dokumentu: událost *onLoad*

`<body onLoad="spustSkript()">`

- řízení událostmi

`<input type="button" onClick="tady je skript"/>`

- spuštění jiným skriptem



Ukázka JavaScriptu

```
2 <html>
3 <head>
4   <meta http-equiv="content-type" content="text/html ;charset=iso-8859-
5   <title>Ukázka JavaScriptu</title>
6   <script type="text/javascript">
7     function dokumentNacten()
8     {
9       alert("Dokument byl načten.");
10    }
11  </script>
12 </head>
13
14 <body onLoad="dokumentNacten()">
15   <h1>Následující text je generovaný skriptem</h1>
16   <script type="text/javascript">
17     document.write("Toto je napsáno pomocí skriptu.");
18   </script>
19   <form onReset="return confirm('Opravdu vymazat obsah formuláře?')">
20     <input type="text" value=""/>
21     <input type="submit" value="Odeslat" />
22     <input type="reset" value="Vymazat"/>
23   </form>
24 </body>
25 </html>
```



Vlastnosti JavaScriptu

■ netypový jazyk

```
var prom = 12;           // prom je Number  
prom = "text";          // prom je String
```

■ datové typy

- String: "řetězec" – řetězec znaků
- Number: 4.5e-12 – libovolné číslo (celé i desetinné; decimální, oktal, hexadec)
- Boolean: true, false – logická hodnota
- Null: null – žádná hodnota
- Object: var pole[] – definován svými vlastnostmi a metodami
- Function: function provedKontrolu() – definice funkce



Vlastnosti JavaScriptu

■ konverze datových typů

```
vysledek = 2 + 3           // vysledek = 5
vysledek = 2 + "3"         // vysledek = "23"
vysledek = 2 + 2 + "3"     // vysledek = "43"
"12" < 3                   // numerické srovnání;false
```

■ pole

- nemají souvislý index, každá položka jiný typ

```
p[0]= 1;                    // p.length==1
p[10]="prvek s indexem 10"; // p.length==11; v paměti 2 prvky
```

- asociativní pole

- metody

```
p.join(,);                  // konverze do String, oddělovač ",",
p.reverse();                // řazení pozpátku
p.sort();                   // alfanumerické řazení
function ciselne_razeni(a,b){return a-b}
p.sort(ciselne_razeni);     // numerické seřazení
```

- build-in pole: např.: forms[], elements[]



Vlastnosti JavaScriptu

■ funkce

– jako datové typy

```
function suma(x,y);  
b = suma; c = b(2,3);  
o = new Object; o.soucet = suma; a = o.soucet(2,3);  
a = new Array(10); a[0]=suma; a[1]= a[0](2,3);
```

– proměnný počet parametrů

```
function suma(){var s=0;  
    for (var i=0;i<suma.arguments.length;i++)  
        s+=suma.arguments[i];  
    return s;}  
soucet1 = suma(2,3);  
soucet2 = suma(5,65,12);
```

Vlastnosti JavaScriptu

■ objekty

- přístup k vlastnostem a metodám

```
document.formular.jmeno           // vlastnost
for (prop in objekt)               //vypíše všechny vlastnosti
    {document.write(objekt[prop]);}
document.write();                 // metoda
```

- objekty se chovají jako asociativní pole

```
document.formular["jmeno"]        // vlastnost
```

- konstruktor, metody, vlastnosti

```
function Obdelnik_plocha() {return this.sirka*this.vyska}
function Obdelnik (s,v) {
    this.sirka=s;this.vyska=v;
    this.obsah = Obdelnik_plocha;
}
obdelnik1 = new Obdelnik(2,3);
o = obdelnik1.obsah();
```



Vlastnosti JavaScriptu

■ Prototypy objektů, třídní proměnné a metody

```
function Kruznice(r) {this.polomer=r}
Kruznice.PI=3.14;
function Kruznice_obsah()
    {return Kruznice.PI*this.polomer*this.polomer}
new Kruznice(0) // takto vznikne instance
Kruznice.prototype.obsah=Kruznice_obsah;
function Kruznice_max(a,b)
    {if (a.polomer<b.polomer) return a else return b}
Kruznice.max = Kruznice_max;

k1 = new Kruznice(2); o = k1.obsah(); x = 1+Kruznice.PI;
k2 = new Kruznice(3); vetsi = Kruznice.max(k1,k2);
```



Vytváření objektů

- Několik možností
 - Objekt Object
 - Constructor funkce
 - Literál objektu
 - Prototyp

Vyrábění instancí pomocí objektu Object

- Třída Object je definovaná v javascriptu defaultně
- Mohu z ní vyrábět instance a těm přidávat vlastnosti dynamicky

```
osoba = new Object();  
osoba.jmeno = "Martin";  
osoba.prijmeni = "Klima";  
osoba.pohlavi = "Muz";  
osoba.bydliste = "Praha";  
  
alert(osoba.jmeno);
```

kuk: objects.html

Vyrábění instancí pomocí funkce

- Vypadá jako normální funkce (a lze ji tak použít)
- Privátní, tj. lokální proměnné jsou uvozené slovem **var**
- Ukazatel na instanci třídy je **this**
- Nový objekt vzniká operátorem **new**
- Jedna funkce může být metodou jiné funkce

```
function X {  
  // definice funkce  
}  
  
instance = new X();
```

Funkce – objekt komplexní ukázka

```
// definice tridy osoba
function osoba (jmeno, prijmeni) {
    this.jmeno      = jmeno;
    this.prijmeni   = prijmeni;
    this.pohlavi    = "Muz";
    // definice metody
    this.nastavPrijmeni = nastavPrijmeni;
    // dalsi definice metody
    this.celeJmeno = celeJmeno;
}
```

Definice třídy, zároveň konstruktor třídy

Přidáme metody, je to ukazatel na jinou funkci

```
// definice metody tridy
// je mimo tuto tridu
function nastavPrijmeni(nove_prijmeni) {
    this.prijmeni = nove_prijmeni;
}

franta = new osoba("Franisek", "Pospisil");
franta.bydliste = "Brno";
franta.nastavPrijmeni("Neruda");
```



Literály

- Literál je daná hodnota
- Můžeme pomocí nich definovat i objekty
- Objekt je vlastně pole dvojic klíč:hodnota, kde klíč je jméno vlastnosti a hodnota její hodnota nebo ukazatel na metodu

```
franta = {  
    jmeno: "Franisek",  
    prijmeni: "Pospisil",  
    pohlavi: "Muz",  
    celeJmeno: celeJmeno};
```

Hodnota

Ukazatel na metodu

```
function celeJmeno() {  
    return this.jmeno+" "+this.prijmeni;  
}  
document.write(franta.jmeno);  
document.write("<br>");  
document.write(franta.prijmeni);  
document.write("<br>");  
document.write(franta.celeJmeno());
```

kuk: objects4_initializer.html

Literály – použití anonymních funkcí

```
franta = {  
    jmeno: "Frantisek",  
    prijmeni: "Pospisil",  
    pohlavi: "Muz",  
    celeJmeno: function () {  
        return this.jmeno+" "+this.prijmeni;  
    }  
};
```

```
document.write(franta.jmeno) ;  
document.write("<br>");  
document.write(franta.prijmeni) ;  
document.write("<br>");  
document.write(franta.celeJmeno()) ;
```

kuk: [objects_anonymni_metody.html](#)



Literály pro pole

- Klasický způsob naplnění pole

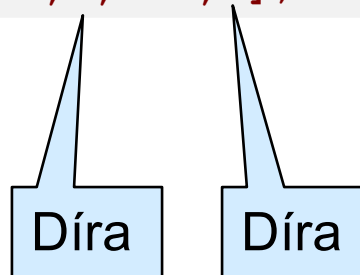
```
pismenka = new Array('a', 'b', 'c');
```

- Pole můžeme naplnit při jeho definici pomocí literálů

```
pismenka = ['a', 'b', 'c'];
```

- Pole můžeme definovat i jako děravé

```
pismenka = ['a', 'b', 'c', , 'e', ];
```



Prototypy

- Prototyp je hodnota, ze které se vytváří instance konkrétní třídy
- Každý objekt má vlastnost **prototype**
- Pamatuje si tím, "z čeho vznikl"
- Nastavení nových vlastností prototypu se projeví na všech instancích, které z něho vznikly



Prototypy

```
// definice tridy osoba
function osoba (jmeno, prijmeni) {
    this.jmeno      = jmeno;
    this.prijmeni    = prijmeni;
    // this.pohlavi   = "Muz";
}

franta = new osoba("Frantisek", "Pospisil");
pepa = new osoba("Josef", "Novak");

// nyní pridej ke vsem instancim tridy osoba tuto vlastnost
osoba.prototype.pohlavi = "Muz";

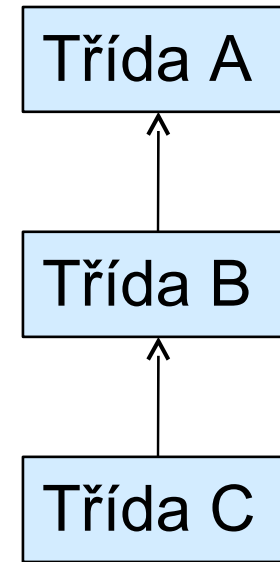
document.write(franta.pohlavi);
document.write("<br>");
document.write(pepa.pohlavi);
```

kuk: objects4_prototype.html



Dědičnost pomocí prototypů

- Chtěl bych "klasickou" dědičnost, tj.
- V javascriptu není rozdíl mezi třídou a instancí
- Finta pomocí prototypů



```
function A() {  
    // definice třídy A  
}  
  
function B() {  
    // definice rozšíření B oproti A  
}  
B.prototype = new A();
```

Dědičnost pomocí funkcí

```
function A(param1) {  
    // definice třídy A  
    this.param1 = param1;  
}  
  
function B(param1, param2) {  
    this.base = A;  
    this.base(param1);  
    this.param2 = param2;  
}  
  
x = new B(100, 200);  
  
document.write(x.param1 + ", " + x.param2);
```

kuk: [objects4_dedicnost.htm](#)

Vzor Singleton v javascriptu

- Singleton je objektový programovací vzor, který umožní udržovat právě jednu instanci objektu v systému
- Problém v javascriptu: nejsou třídy, nejsou třídní proměnné
- Finta: využijeme tzv. anonymní třídu
- Třída (instance) je definovaná v okamžiku jejího vzniku

```
singleton_object = new function Singleton() {  
    // definice tridy zde  
}
```

Abych řekl pravdu,
tak to docela
pokulhává

Singleton ukázka

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> Singleton </TITLE>
</HEAD>

<script type="text/javascript">

// definice tridy Singleton a její instanciovani
singlePocitadlo = new function Pocitadlo () {
    this.hodnota    = 0;
}

// pouziti
singlePocitadlo.hodnota++;
document.write (singlePocitadlo.hodnota) ;

</script>
<BODY>

</BODY>
</HTML>
```

Funkci nemusím
pojmenovat, ale je
to lepší pro ladění

kuk: vzor_singleton.html



Singleton pokr

- Přidáme nějaké metody
 - opět jako anonymní (třídy nebo instance??)

```
// definice tridy Singleton a její instanciovani
```

```
singlePocitadlo = new function Pocitadlo() {
```

```
    this.hodnota = 0;
```

```
    this.inc = function inc() {
```

```
        this.hodnota++;
```

```
    }
```

```
    this.getHodnota = function h () {
```

```
        return this.hodnota;
```

```
    }
```

```
}
```

```
// pouziti
```

```
singlePocitadlo.inc();
```

```
singlePocitadlo.inc();
```

```
document.write(singlePocitadlo.getHodnota());
```

Funkci mohu pojmenovat
ale nemusím

kuk: vzor_singleton2.html



Děkujeme za pozornost



Computer Graphics Group



Práce s cookies

- Je poměrně nepříjemná
- Je potřeba "naparsovat" řetězec

- Viz ukázka `cookie1.html`



FAQ

- jak zamezit odeslání špatně vyplněného formuláře
`<form onsubmit="return validuj()">`
kde funkce validuj vraci true nebo false
- jak najít element podle jeho ID?
`document.getElementById(id_elementu)`
pozor: element nemusí být nalezen => ošetřit
- jak nehledat element podle ID?
`document.all["ID"]`
toto je proprietární řešení pro IE

FAQ

- jak otevřít nové okno?

```
nove_okno = window.open(url);
```

pozor, měl bych si zapamatovat odkaz na toto okno, abych mohl např. udělat toto:

```
nove_okno.focus();
```

- jak vytvořit nový element a přidat (debrat) ho z/do dokumentu?

```
oOption = document.createElement("OPTION");
```

```
nejaky_element.appendChild(oOption);
```

```
nejaky_element.removeChild(oOption);
```



FAQ

- Jak nastavit atribut?

`nejaky_element.nejaky_atribut = "hodnota";`

nebo

`nejaky_element.setAttribute("nejaky_atribut", "hodnota");`

