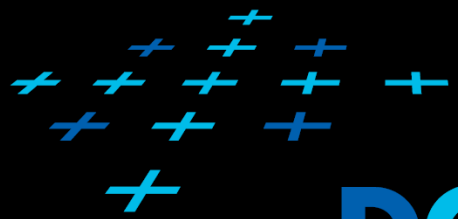


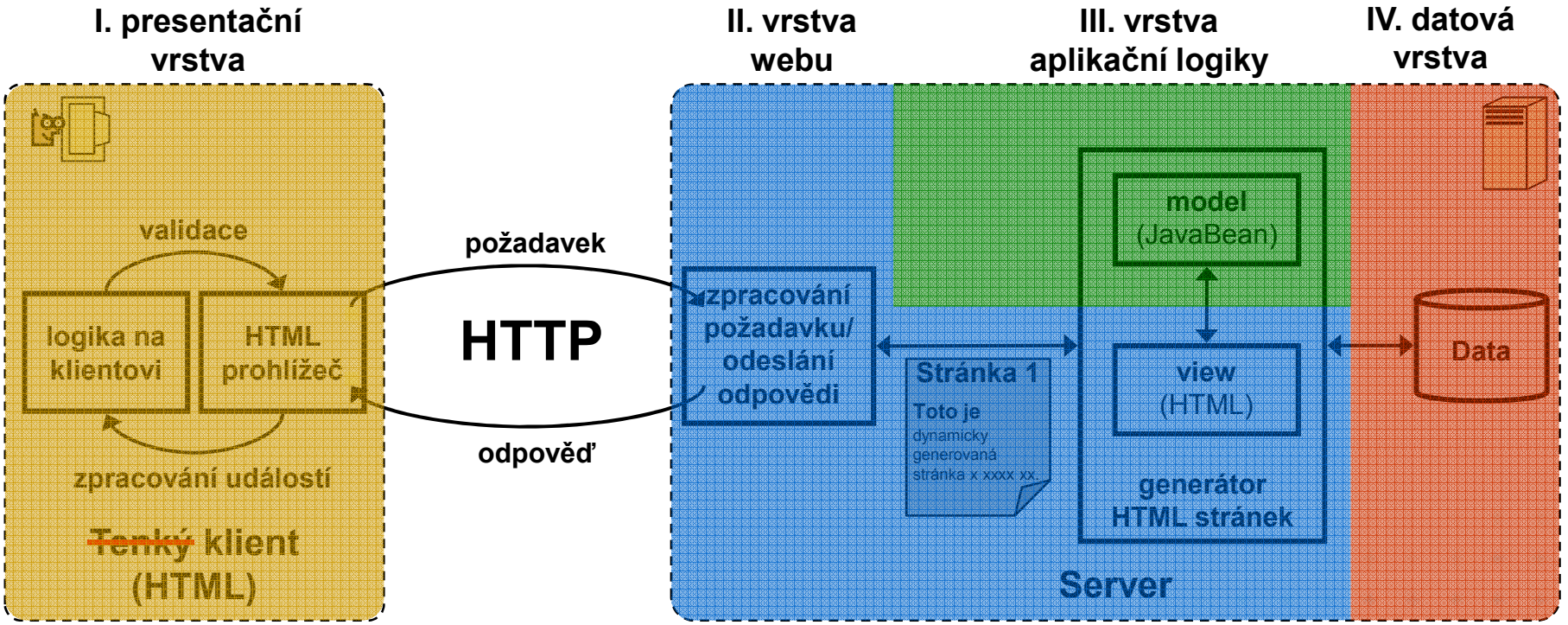
DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Logika na straně klienta, skriptovací jazyky

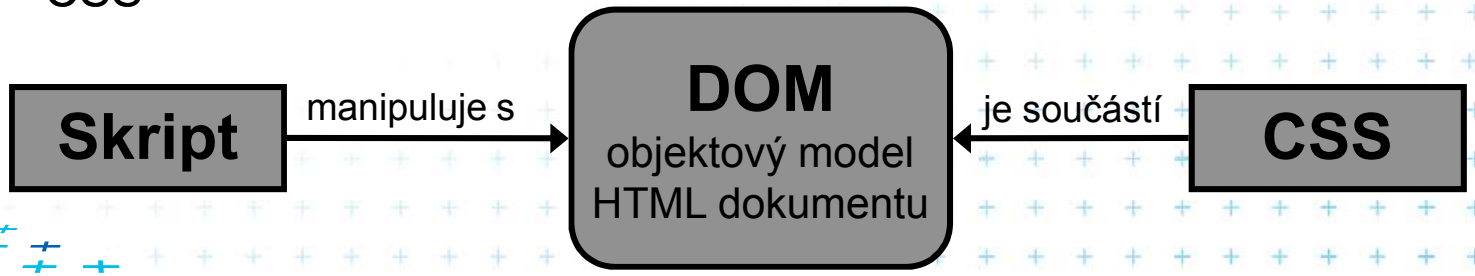
Martin Klíma

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



DCGI

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
  - bezpečností důvody

K čemu skripty slouží a k čemu ne?

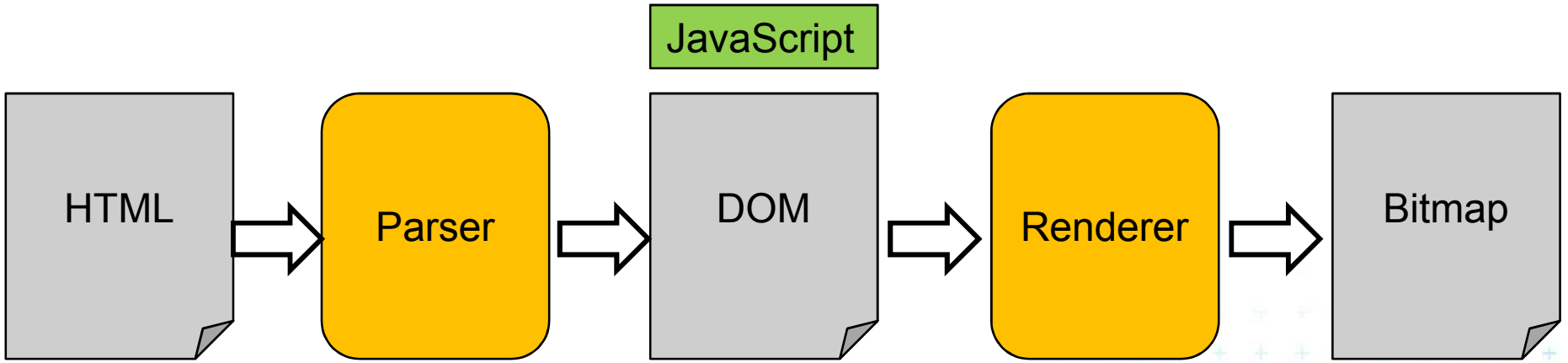
- ANO
  - kontrola a předzpracování vstupních dat (formuláře)
  - manipulace s malými objemy dat
  - dymické změny obsahu HTML
    - událost => změna HTML elementu (např. obrázků, položek ve formuláři), generování HTML do nových oken prohlížeče
- NE
  - spouštění aplikací na klientském počítači
  - manipulace se soubory a adresáři

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

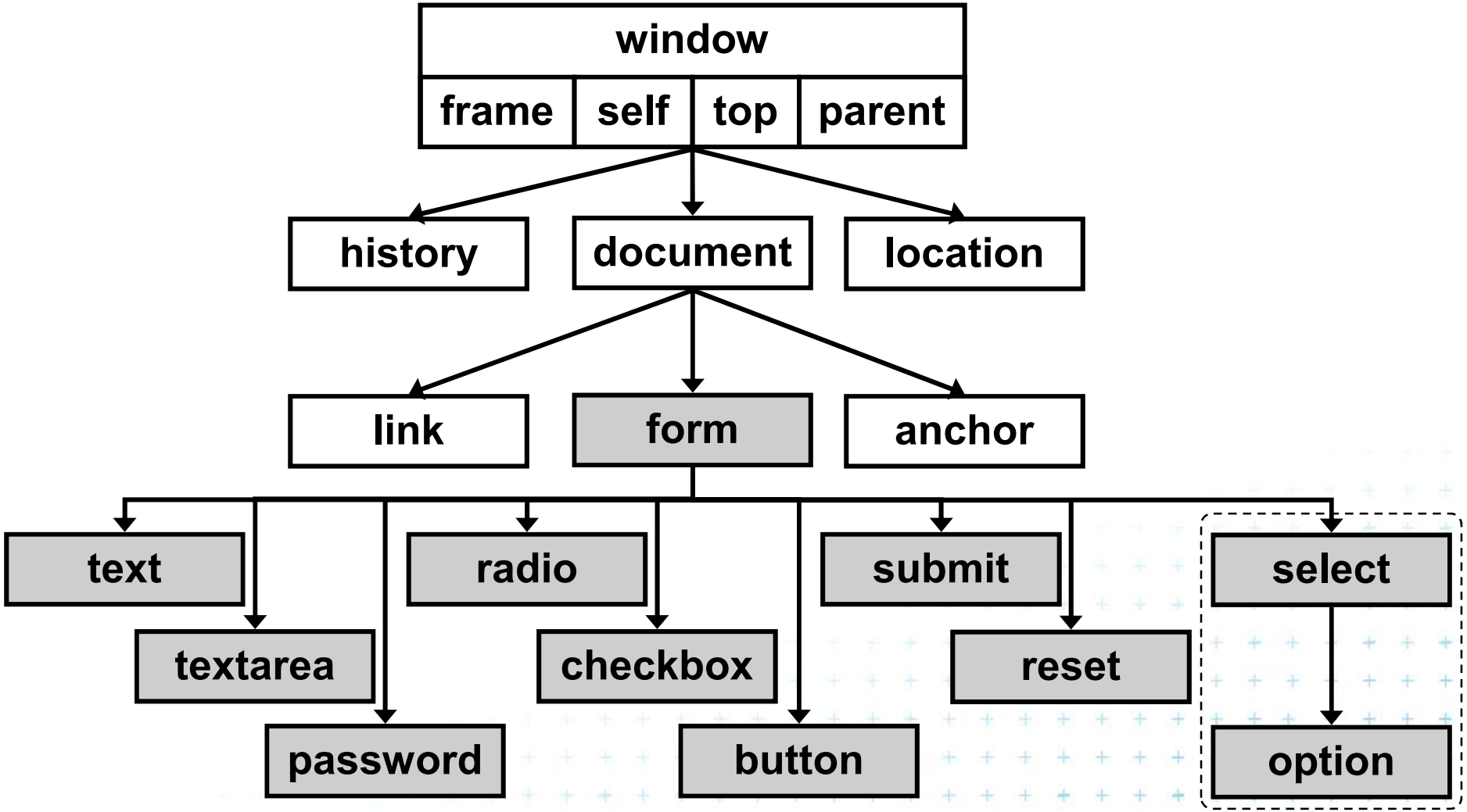
Vývoj DHTML

- 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

Zpracování dokumentu



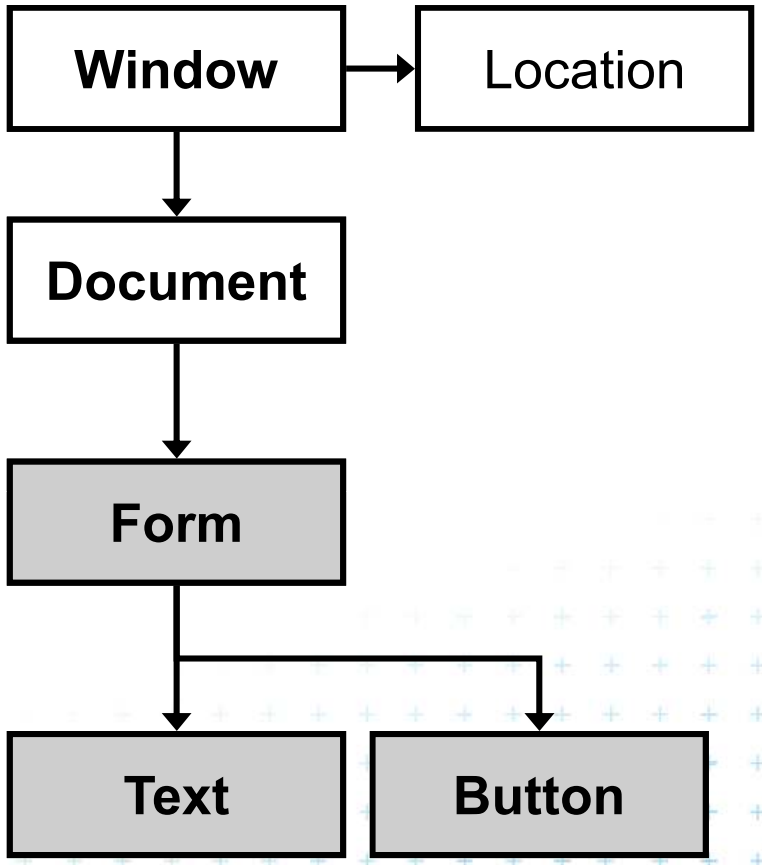
DOM - hierarchie



DCGI

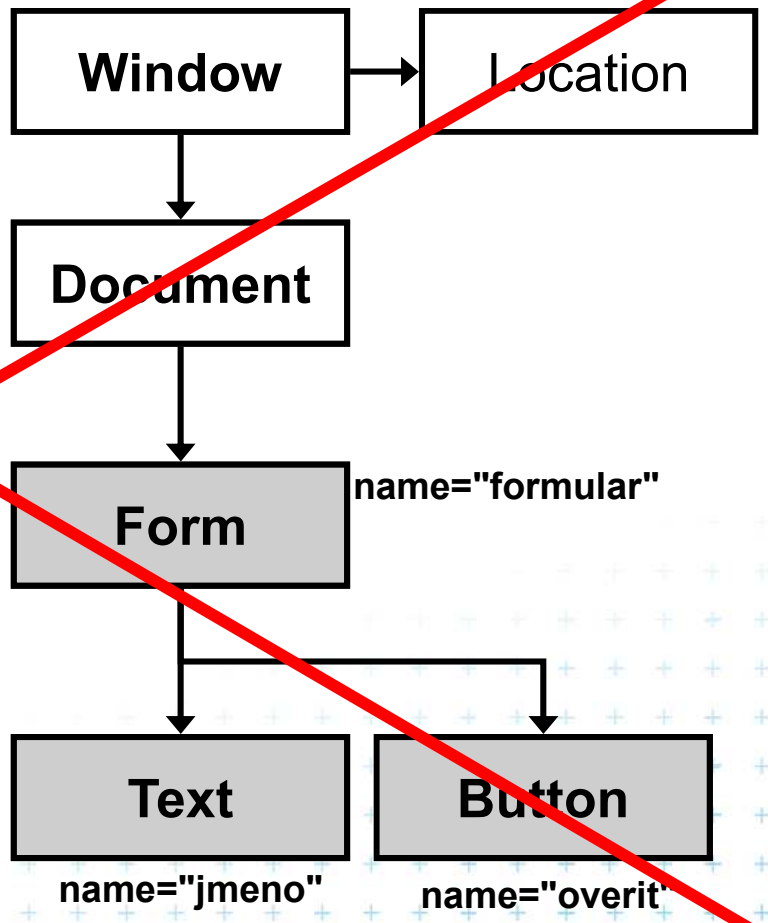
DOM: Ukázka

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



DOM: reference

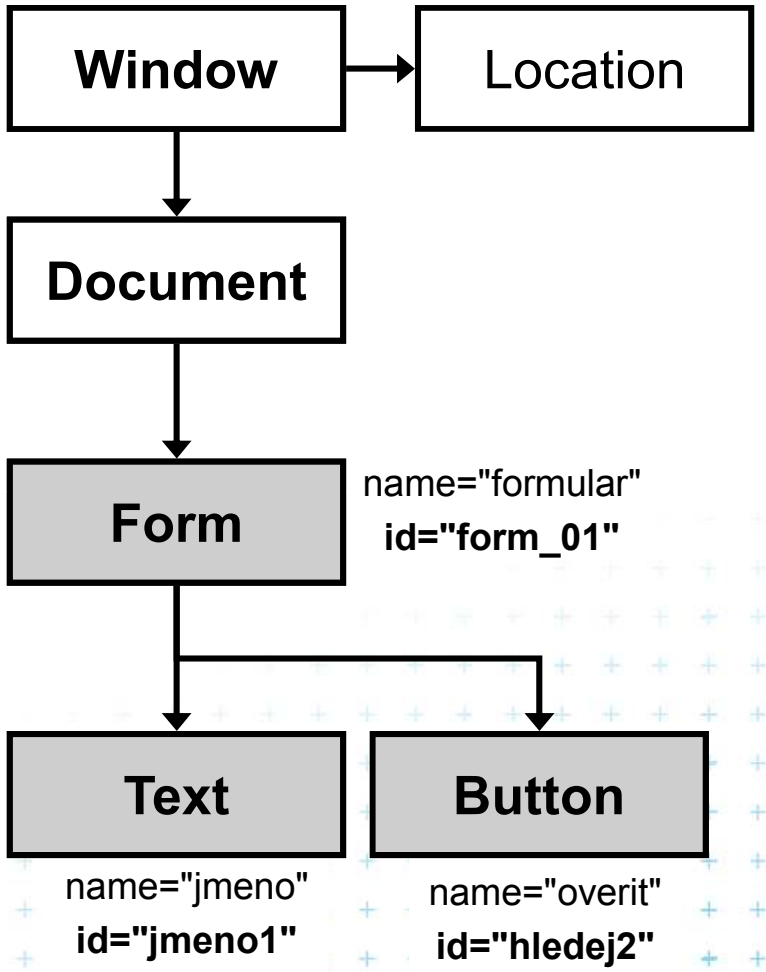
- window
- window.document
- window.document.formular
- window.document.forms[0]
- window.document.forms["formular"]
- window.document.formular.jmeno
- window.document.formular.elements[0]
- window.document.formular.elements["jmeno"]
- window.document.formular.overit
- window.document.formular.elements[1]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



DOM: reference

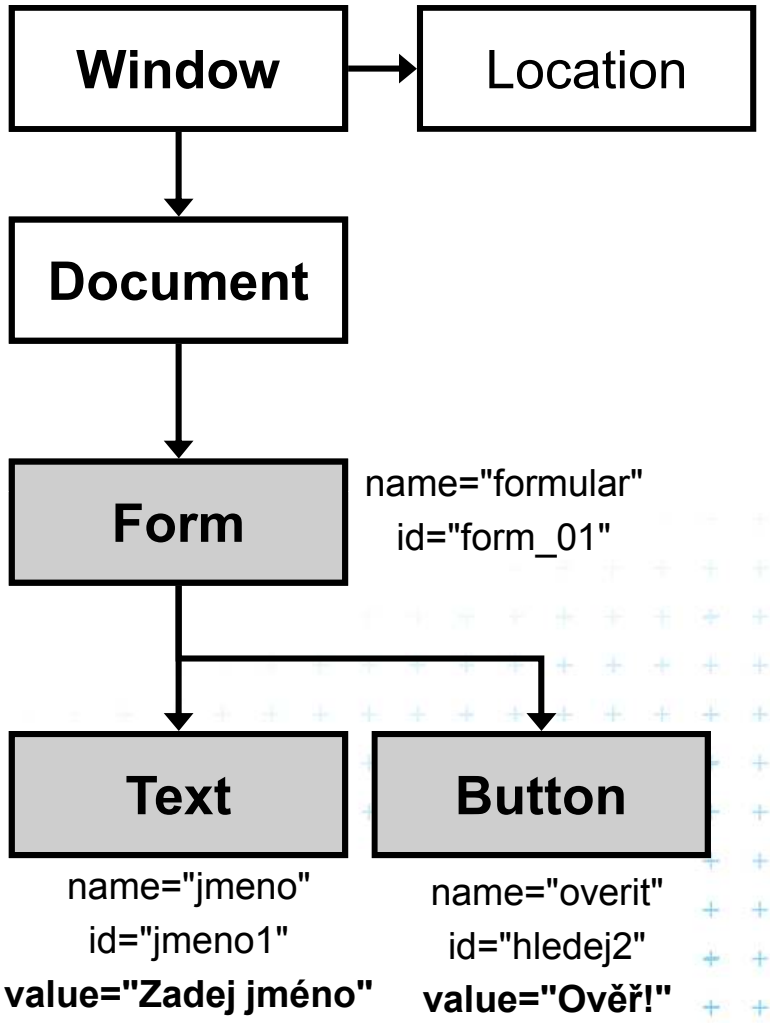
- document.getElementByID("form_01")

- document.getElementByID("jmeno1")
- document.getElementByID("hledej2")



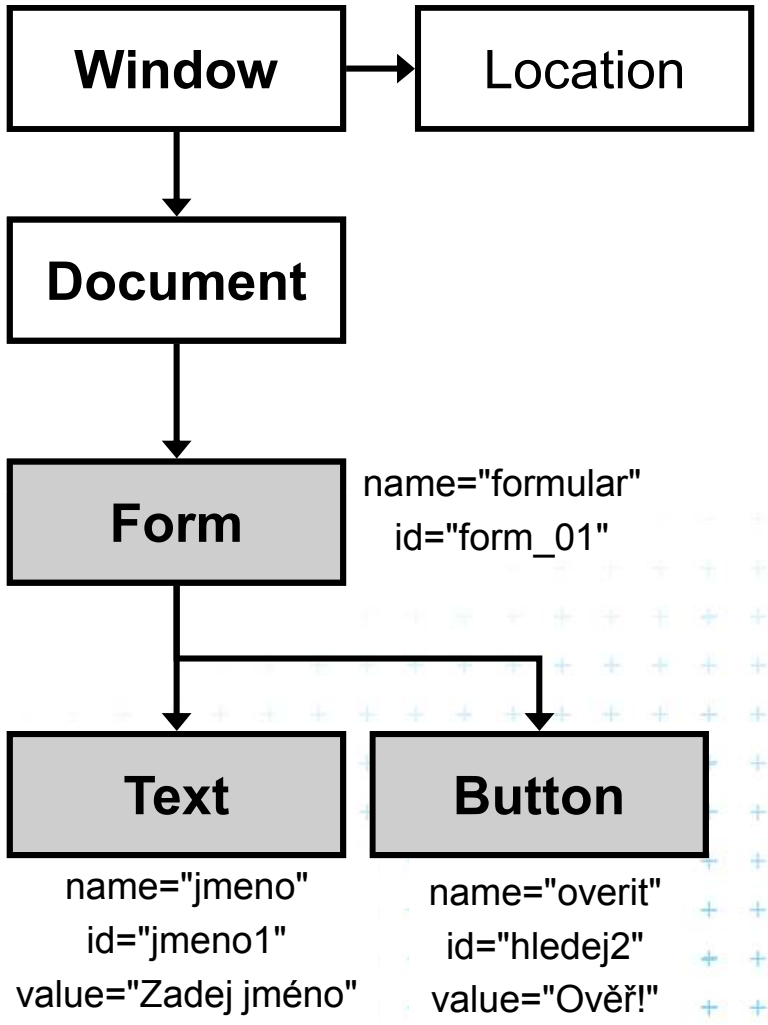
DOM: Vlastnosti (properties)

- document.getElementByID(“jmeno1").value



DOM: Metody

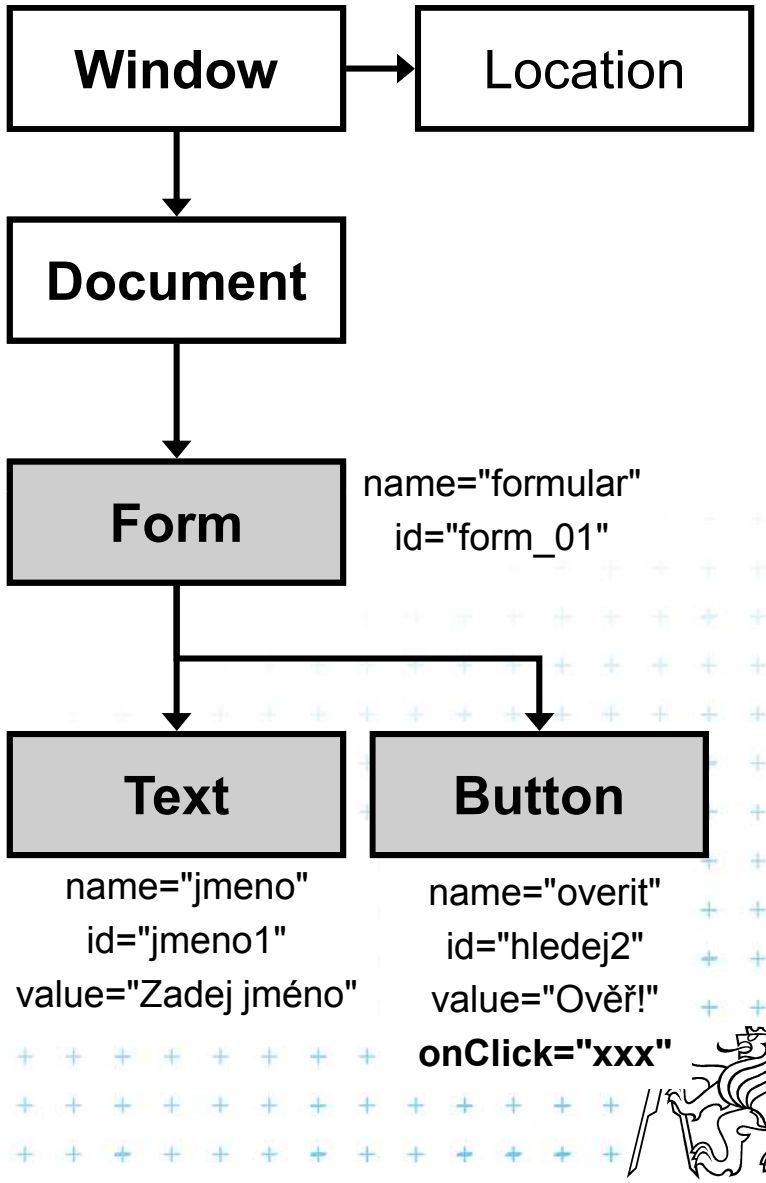
- window.moveTo(30,50)
- ~~document.write("Nějaký text")~~
- document.getElementByID (“form_01”).submit()
- document.formular.jmeno.select()



DCGI

DOM: Ovladače událostí

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



DCGI

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"/>
```

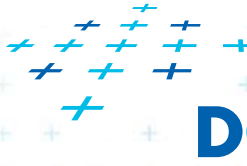
Skripty a HTML: kam ho zapsat

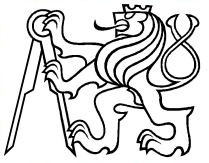
```
<html>
   <head>
      <title>Jednoduchý dokument</title>
      <script type="text/javascript">tady je skript</script>
   </head>
   <body>
      <h1>Tělo dokumentu</h1>
      <script type="text/javascript">tady je skript</script>
      <form>
            <input type="text"/>
            <input type="button" onClick="tady je skript"/>
      </form>
   </body>
</html>
```

reakce na události

tvorba obsahu při načítání

reakce na události





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>
```

DCGI

Vlastnosti JavaScriptu

- proměnné

```
var prom; // deklarace, lokální prom.
prom2 = "ahoj";  // deklarace a definice, globální prom.
```

- 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 definován svými vlastnostmi a metodami
  - Function: function provedKontrolu() – definice funkce
  - Undefined

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
```

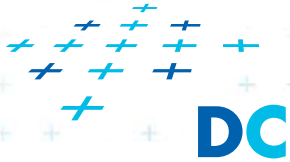
- pole
  - nemají souvislý index, každá položka jiný typ

```
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);
```

  - funkce může mít přiřazeny proměnné
    - fungují jako statické proměnné

Javascript – pokročilé programování

- Vytváření objektů
- Prototypy a dědičnost
- Události
  - Event Handler
  - Event Objekt
  - Probublávání
  - Ošetřování chyb

V čem se Javascript liší od např. Javy

- Javascript má jiné pojetí objektů
- Neexistuje zde klasická hierarchie tříd
- Třídy vlastně neexistují
- Existují jen instance a tzv. PROTOTYPY

Klasika

Class

Instance

Prototypovací jazyk

Class, Instance jedno jest

Vytváření objektů

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

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 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");
```

Definice třídy, zároveň konstruktor třídy

Přidáme metody, je to ukazatel na jinou funkci

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};

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());
```

Hodnota

Ukazatel na metodu

kuk: objects4_initializer.html

Literály – použití anonymních funkcí

```
franta = {
        jmeno:"Franisek",
        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

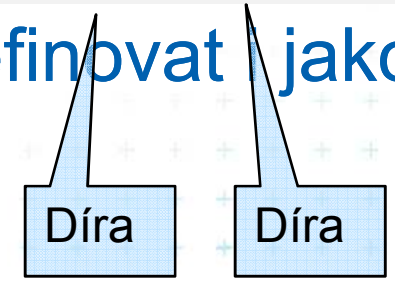
```
pismenka = new Array('a', 'b', 'c');
```

- Pole můžeme naplnit při jeho definici pomocí literálů

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

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

- Pole můžeme definovat i jako děravé



UDÁLOSTI

DCGI

Události v javascriptu

- Implicitní definice Event-handleru
- Explicitní definice Event-handleru
- Objekt event
- Životní cyklus události
- Probublávání události

Události

- Události jsou generovány v uživatelském rozhraní
- Máme možnost je odchytnout a napojit na nějaký vlastní kód
- Část programu, která ošetřuje události se nazývá **Event-Handler**

Ošetření událostí

- Některé události mají přiřazené implicitní akce
- Tyto akce jsou volány, pokud neřekneme jinak
- Příklad:
  - click na odkazu způsobí přechod na jinou stránku
  - click na tlačítko submit způsobí odeslání formuláře

- Jestliže definujeme vlastní akci, je pořadí vykonání
  1. vlastní definované akce
  2. implicitní akce

Zrušení default akce

- w3c: `event.preventDefault()`
- MS: `event.returnValue = false;`

nebo

```
return false;
```

kuk: prevent_default_udalosti.html

Defaultní akce - Výjimka

- je mouseover a **`window.status`**
- Stornuje se pomocí

```
return true;
```

Nikdo neví proč :-(

Napojení na události

- Inline registrace
- Tradiční registrace
- Pokročilá registrace

Inline registrace událostí

```
<A HREF="somewhere.html"
                onClick="alert('I\'ve been clicked!')">

<A HREF="somewhere.html" onClick="doSomething()">
```

- Registrace je napsaná jako součást HTML kódu
- Není to moc elegantní
- Nedoporučuje se, protože se míchá html a javascript

kuk: events_register_inline.html

Programová ragistrace událostí

- Registrace

```
element.onclick = doSomething;
```

- Zrušení registrace

```
element.onclick = null;
```

Programové vyvolání události

- je to normální funkce
- lze ji zavolat z programu

```
element.onclick()
```

- Pozor, špatná implementace v IE 5.5

```
element.fireEvent('onclick')
```

Spouštění funkce

- Handleru události předáme ukazatel na funkci

```
function doSomething() {
        alert ("Stala se událost");
}
```

```
element.onclick = doSomething
```

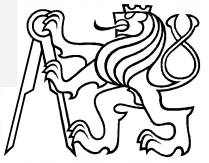
Toto je ukazatel na funkci, žádné závorky!

Event handler ukázka

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <title>
            Mouse Capture
        </title>
        <script type="text/javascript">
        function action() {
         alert("udalost odchycena");
        }

        </script>
    </head>
    <body onload=
    "document.getElementById('div1').onclick = action;">
        <div id="div1" style=
        "height:100;width:200;background-color:red">
            Zde je klikaci prlocha
        </div>

    </body>
</html>
```



Jak registrovat více obsluh k jedné události?

- Co nefunguje:

```
element.onclick = doSomething;
element.onclick = doSomethingElse;
```

Druhá registrace přepíše první ☹

- Co funguje:
  - použijeme anonymní funkci

```
element.onclick =
function(){doSomething(); doSomethingElse();}
```

Registrace Event-handlerů podle standardu W3C

```
element.addEventListener('click',
                doSomething, false);

element.addEventListener('click',
                doSomethingElse, false);
```

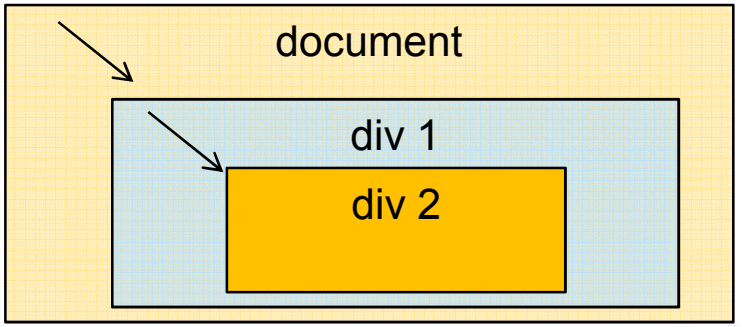
- Registrují se oba
- Poslední argument určuje, zda se událost má odchytit ve fázi capture nebo bubble (false=bubble)

```
element.removeEventListener('click', doSomethingElse, false)
```

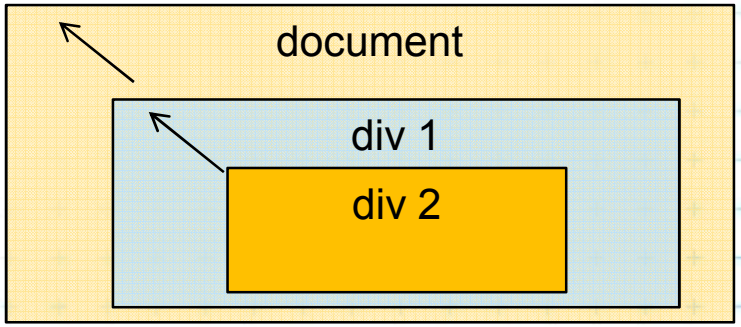
- Odstranění Event-handleru

Způsoby zachytávání událostí

- Otázka: jestliže mám vnořený element který odchytává stejnou událost jako jeho nadřazený element, kdo to má odchytit první?
- Event Capture
  1. document
  2. div 1
  3. div 2
- Event Bubbling
- div 2
- div 1
- document



Netscape

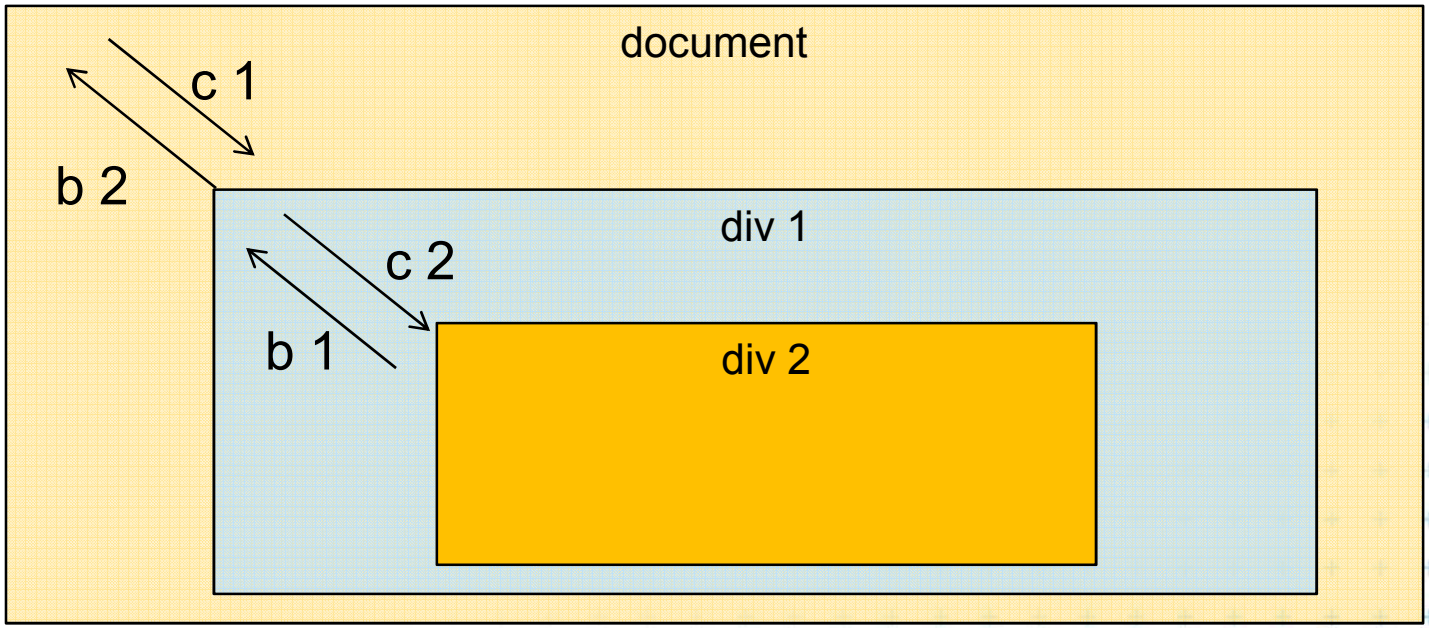


Microsoft

DCGI

Propagovaní události – W3C

- Kombinace capture a bubble propagace
- Nejprve capture (c), pak zpětné bubble (b)



kuk: capture.htm

Registrace událostí v IE

Registrace

```
element.attachEvent('onclick', doSomething)
element.attachEvent('onclick',doSomethingElse)
```

Zrušení registrace

```
element.detachEvent('click', doSomething)
```

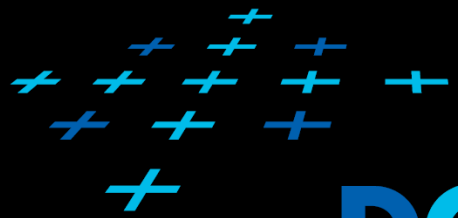
Zrušení probublávání

- Standard

```
event.cancelBubble = true;
```

- IE

```
event.stopPropagation();
```



DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Děkuji za pozornost