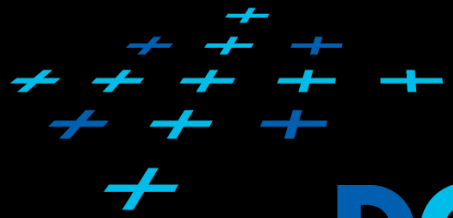


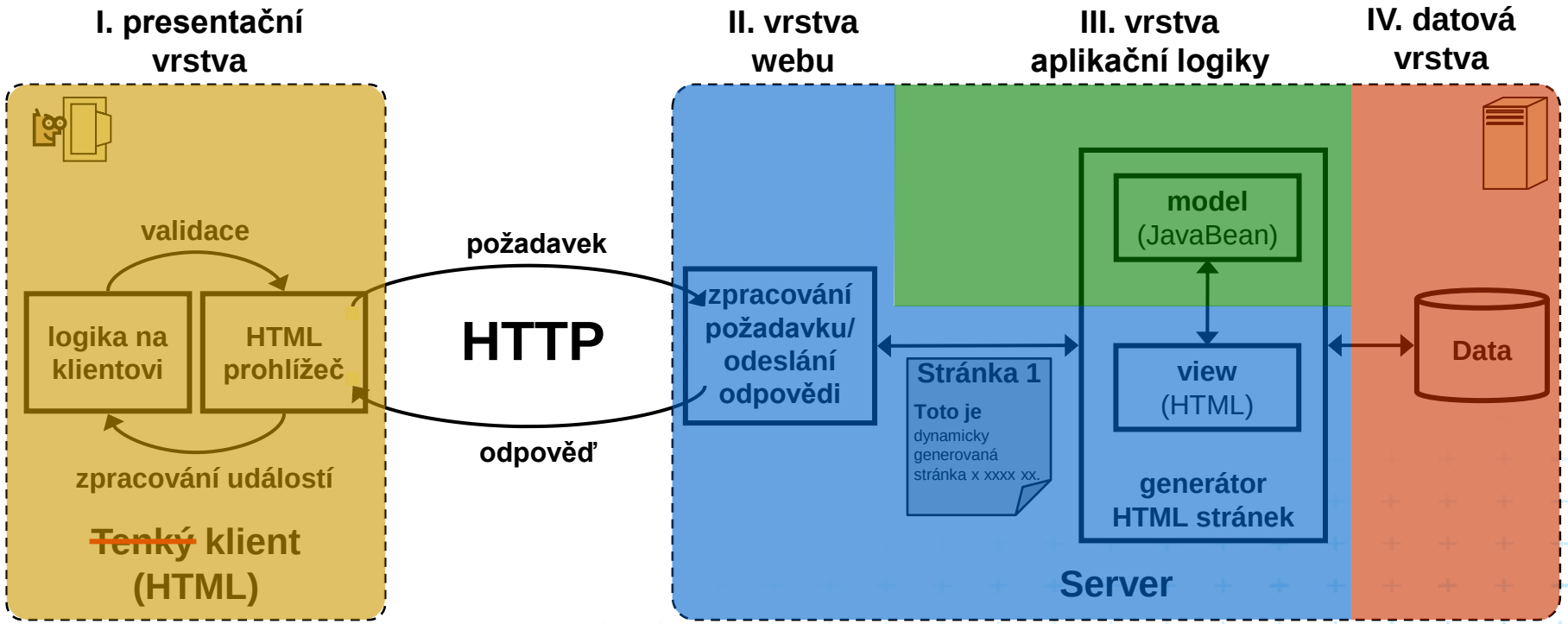
DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Logika na straně klienta

Martin Klíma

Architektura web aplikace: dynamický web



Charakteristika JavaScriptu

- skriptovací jazyk
  - interpretovaný klientem
  - nezávislý na platformě (nutné pro použití všude)

- skripty v prohlížeči pracují v definovaném prostředí
  - omezené možnosti (sandbox)
  - datový model DOM
  - UI+prezentace dat: řeší HTML prohlížeč
  - události

Historie a verze

- V historii vzniklo několik různých interpreterů = engine
- Pěkná tabulka z Wikipedie:

Scripting engine	Reference application(s)	Conformance			
		ES5[	ES6 (2015)	ES7 (2016)	Newer (2017+)
SpiderMonkey	Firefox 94	100%	98%	100%	100%
V8	Google Chrome 95, Microsoft Edge 95, Opera 80	100%	98%	100%	100%
JavaScriptCore	Safari 15	100%	99%	100%	90%

Node JS

Co JavaScript v prohlížeči smí a co ne

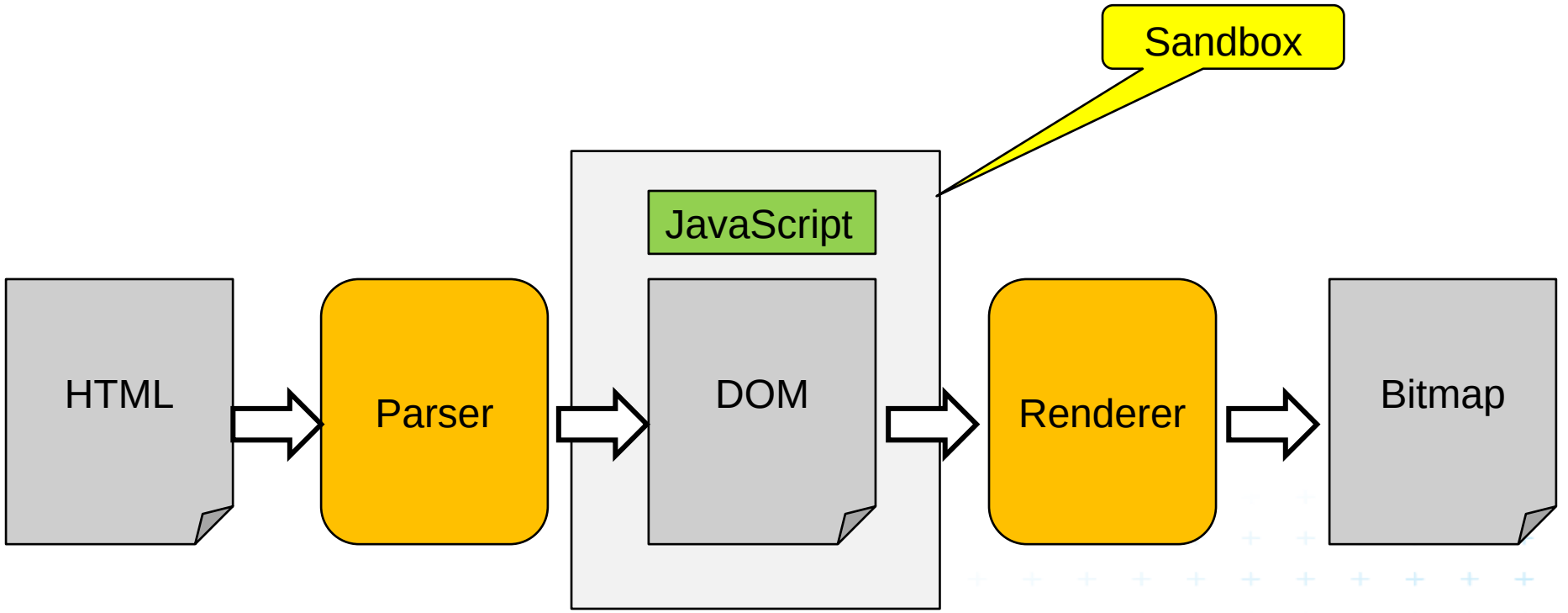
- Smí
  - Měnit DOM
  - Dělat HTTP dotazy
  - Odesílat soubory
    - jen ty, které explicitně určí uživatel
  - Pracovat s
    - kamerou
    - mikrofonem
    - pozicí
    - orientací zařízení
  - Měnit UI – ukazatel kurzoru
- Nesmí
  - Odesílat libovolné soubory
  - Pracovat se zdroji mimo prohlížeč
  - Pracovat se sítí
    - jiné, než HTTP protokoly
    - otevírat porty
  - Měnit nastavení počítače

Pozor!

Jmenovaná omezení se aplikují jen na běhové prostředí JavaScriptu uvnitř webového prohlížeče.

JavaScript jako takový umí vše, je to plnohodnotný programovací jazyk jako spousta jiných.

Zpracování HTML dokumentu



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 scripty spouští

- při načítání dokumentu okamžitě

```
<body>
   <h1>Tělo dokumentu</h1>
   <script type="text/javascript">tady je script</script>
<body>
```

- řízení událostmi

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

Takto to ale neděláme :-)

- spuštění jiným scriptem

DCGI

JavaScript

SYNTAXE A ZÁKLADY JAZYKA

Proměnné a konstanty

4 typy definice proměnných

typ	chování	rozsah platnosti
automatický	definice bez deklarace	globální
var	deklarace	ve funkci
let	deklarace	v lokálním bloku
const	deklarace s definicí, konstanta	v lokálním bloku

```
a = "Ahoj"; // automatická globální deklarace a přiřazení

var b; // deklarace ve scope funkce

b = 200; // definice hodnoty

let c = "Praha";// deklarace s definicí, scope blok

const d = "Brno"; // deklarace s definicí, scope blok, nelze změnit

d = "Ostrava"; // error
```

Uncaught TypeError: Assignment to constant variable.

Hoisting = vytáhnutí dopředu

- Deklarace proměnné před jejím použitím se povoluje
- Přiřazení hodnoty se ale neaplikuje

```
var x = 5; // Initialize x
var y = 7; // Initialize y

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y
```

```
5 7
```

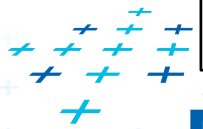
Hoisting

```
var x = 5; // Initialize x
var y;
elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y

var y = 7; // Initialize y
```

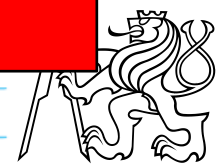
```
5 undefined
```

```
var x = 5; // Initialize x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y
```

```
script.js:5 Uncaught
ReferenceError: y is not defined
```



DCGI

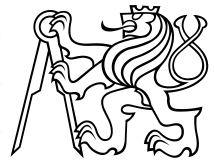


Co z toho plyne?

Všechny proměnné vždy deklarujeme na začátku scriptu!!!

Datové typy

- Základní
  - string – “ahoj” nebo ‘ahoj’
  - number – interně vždy double
  - bigint – int pro velké hodnoty
  - boolean – true/false
  - undefined – deklarovaná proměnná bez hodnoty
  - null – prázdná hodnota
  - symbol – unikátní jednoduchá hodnota
  - object – sada pojmenovaných hodnot
  - function – ukazatel na funkci
- Složité
  - Object – sada pojmenovaných hodnot
  - Array – sada pojmenovaných hodnot
  - Date – datum ☺
  - String – objekt typu String
  - Number – objekt typu Number
  - Boolean – objekt typu Boolean



Datové typy - přetypování

Dynamické typování ☹

```
let time; //undefined
time = 10.5; //number
time="Poledne"; //string
```

```
// data conversions
let r1 = 20 + " let";          // "20 let"
let r2 = "let " + 20;          // "let 20"
let r3 = "20" + " let"         // "20 let"
let r4 = 16 + 4 + " let";      // "20 let"
let r5 = "let " + 16 + 4;      // "let 164" – POZOR
let r6 = "5" - 2;              // number 3
let r7 = "5" * "2";            // number 10
```

Kompletní tabulka převodů typů je zde:
https://www.w3schools.com/js/js_type_conversion.asp

Převody typů

```
Number("3.14");          //number 3.14
Number(Math.PI);         //number 3.141592653589793
Number(" ");             //number 0
Number("");               //number 0
Number("blabla");        //number NaN, hodnota pro necislo
Number(false);           //number 0
```

Unární operátor +

```
let y = "5";   // string "5"
let x = + y;   // vynutíme převod na number, number 5
```

```
// cislo na string
String(x);          // string hodnota x
x.toString();       // to samé
String(123);        // "123"
123.toString();     // to samé
String(100 + 23);   // "123"
(100+23).toString(); // to samé
```

DCGI

Pole - Array

Je to sada hodnot ve formě klíč – hodnota

Pokud „neřešíme“ klíče, jsou položky číslovány automaticky

```
const pole1 = ["Pondělí", "Úterý", "Středa"]; //const je ukazatel na to stejné pole, jeho položky ale měnit můžeme
pole1.push("Čtvrtek"); // použití metod pole, výsledek ["Pondělí", "Úterý", "Středa", "Čtvrtek"]

const pole2 = new Array("Sever", "Jih", "Východ", "Západ"); // jiný způsob vytváření polí
```

Metody nad poli

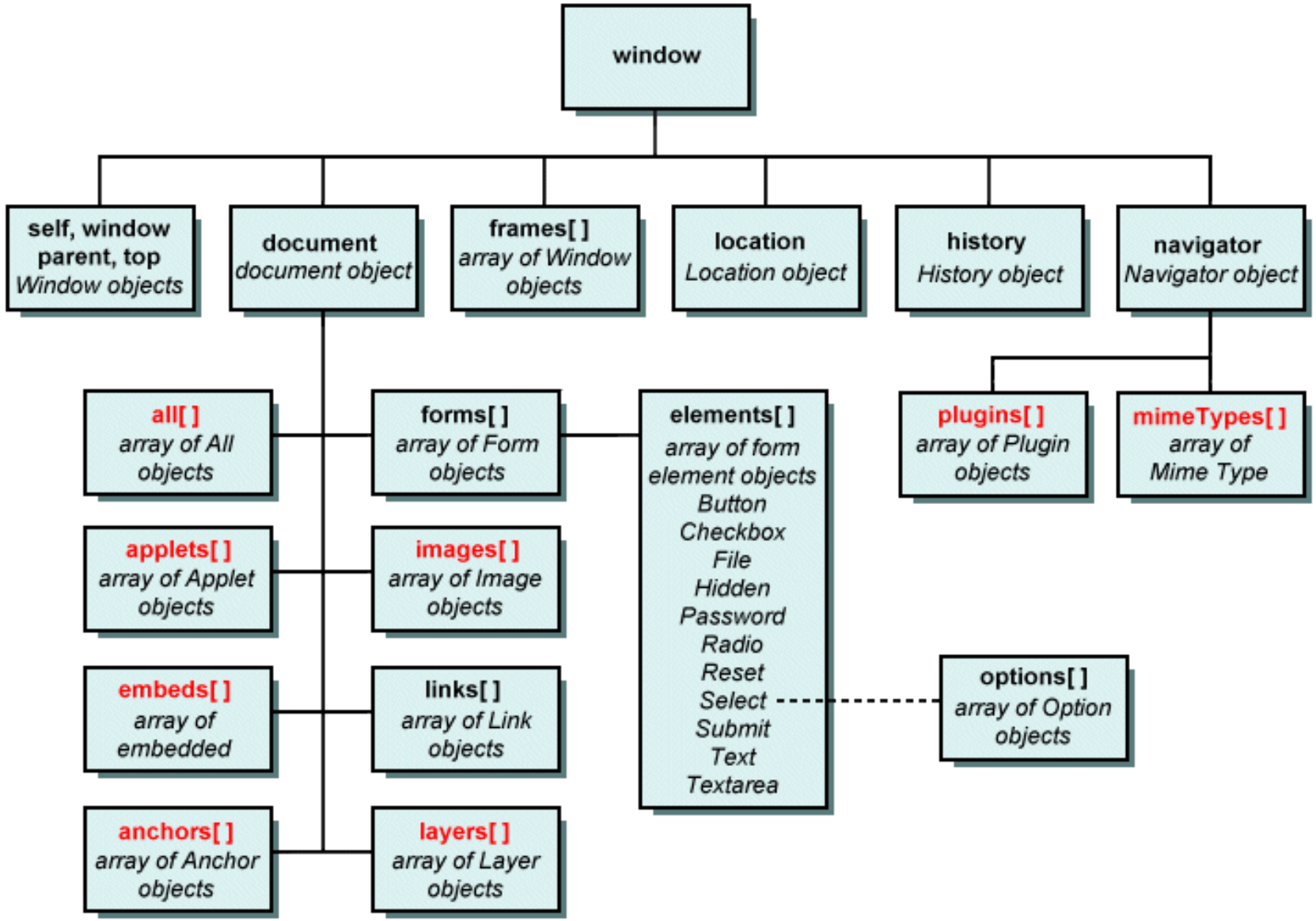
Array length
Array toString()
Array pop()
Array push()
Array shift()
Array unshift()

Array join()
Array delete()
Array concat()
Array flat()
Array splice()
Array slice()

DCGI

Propojení s prohlížečem

- reference: https://www.w3schools.com/jsref/default.asp



zdroj: https://medium.com/@reettikgoswami97/document-object-model-dom-c19d66abd235

DOM

- Objekty document a window jsou vždy přítomny
- document reprezentuje nahraný html dokument
- vlastnosti jsou dostupné přes tečkovou notaci
- některé vlastnosti jsou jen pro čtení, jiné i pro zápis

window.document

```
let doc = window.document;                  // reference na document
let doc2 = document;                        // zjednodušená notace, stejný výsledek

let p1 = doc.getElementById("p1");          // vrátí ukazatel na element s id="p1" nebo null
if (p1 != null) {                           // nalezeno?
        window.console.log("element p1 nalezen");
} else {
        console.log("Odstavec p1 nenalezen."); // zjednodušená notace
}
```

Objekt document

- mnoho metod, viz
  https://www.w3schools.com/jsref/dom_obj_document.asp

createElement

```
const para = document.createElement("p");
para.innerText = "Obsah noveho odstavce";
document.body.appendChild(para);
```

createEvent

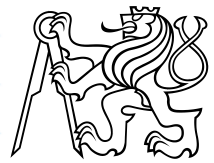
```
const ev = document.createEvent("MouseEvent");
ev.initMouseEvent("mouseover", true, true, window, 0, 0, 0, 0, 0,
false, false, false, false, 0, null);
document.getElementById("myDiv").dispatchEvent(ev);
```

DCGI

Objekt document II

- getElementById() – vrací jeden nebo null
- getElementsByClassName() – vrací pole
- getElementsByName() – vrací pole
- getElementsByTagName() – vrací pole

```
// zaskrtneme vsechny checkboxy, ktere maji atribut name="animal"
const collection = document.getElementsByName("animal");
for (let i = 0; i < collection.length; i++) {
  if (collection[i].type == "checkbox") {
    collection[i].checked = true;
  }
}
```



Objekt document III

querySelector – vrací první výskyt elementu, který pasuje na css selektor

```
// oznacime prvni krasny odstavec tricou "oznaceny"
let  odstavec = document.querySelector("p.krasny");
odstavec.innerHTML = "Krásný odstavec";
odstavec.classList.add("oznaceny");
```

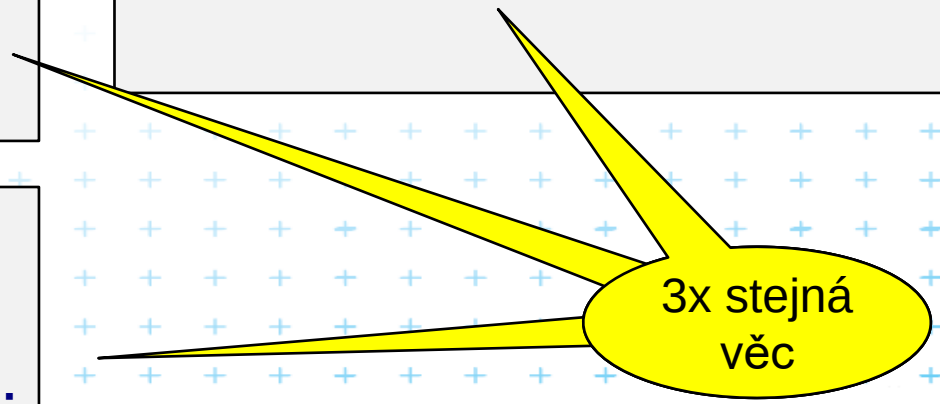
```
p.oznaceny {
        border: 1px solid black;
        background-color: yellow;
}
```

querySelectorAll

```
// najdeme vsechny odstavce a pridame jim na zacatek obsahu *
const list = document.querySelectorAll("p");
for (let i = 0; i < list.length; i++) {
  list[i].innerHTML = "* " + list[i].innerHTML;
}
```

```
document.querySelectorAll("p").forEach((el) => {
        el.innerHTML = "* " + el.innerHTML;
})
```

```
function addAsterix (el, index) {
        el.innerHTML = "* " + el.innerHTML;
}
document.querySelectorAll("p").forEach(addAsterix);
```



Funkce a ukazatel na funkci

```
// definice funkce
function secti (a, b) {
        return a + b;
}

var vysledek = secti (10, 20); // 30
console.log (vysledek);
```

```
// inline funkce
// list vsech odstavcu
const list = document.querySelectorAll("p");

list.forEach(function (element) {
        element.innerHTML = "nasel jsem element p";
});
```

```
// ukazatel na funkci
function nasel (element) {
    element.innerHTML = "nasel jsem element p";
}
document.querySelectorAll("p").forEach(nasel);
```

```
// inline funkce 2
document.querySelectorAll("p").forEach((element)=> {
        element.innerHTML = "nasel jsem element p";
});
```

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

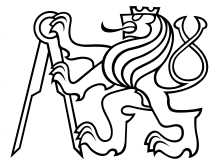
Registrace Event-handlerů podle standardu W3C

```
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)

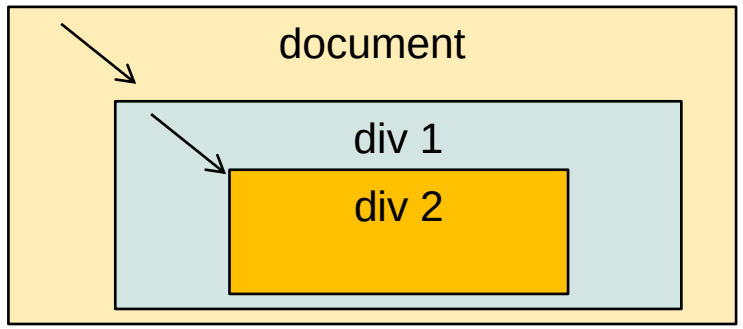
- Odstranění Event-handleru

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

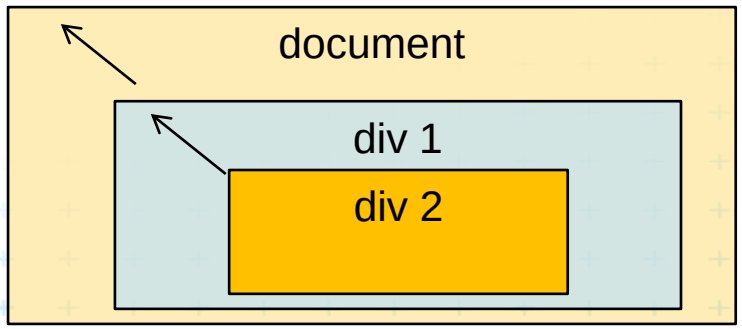


Způsoby zachytávání událostí

- 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



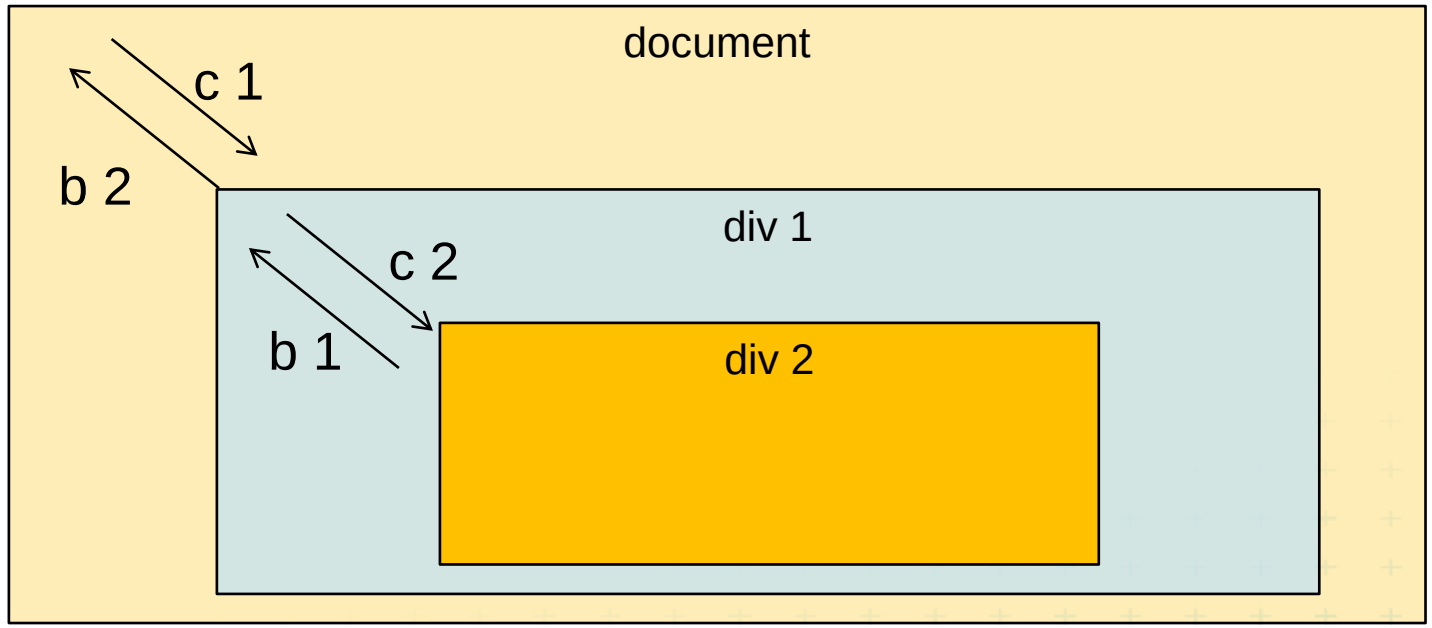
Netscape



Microsoft

Propagovaní události – W3C

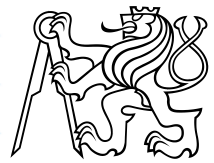
- Kombinace capture a bubble propagace
- Nejprve capture (c), pak zpětné bubble (b)



DCGI

Události - příklad

```
var p = document.querySelector("p");

var posluchac = function() {
    alert("kliknuto");
}

p.addEventListener("click", posluchac); // předáno odkazem

/* ... */

p.removeEventListener("click", posluchac);
```



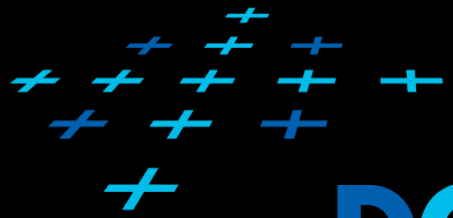
Event objekt

```
document.body.addEventListener("click", function(e) {
    alert(e.type);      // "click"

    alert(e.clientX);   // souřadnice kurzoru
    alert(e.target);    // ?

    alert(e.ctrlKey);   // informace o klávesách
});
```

DCGI

Výchozí akce a zastavení akce

```
var form = document.querySelector("form");
form.addEventListener("submit", function(e) {
    e.preventDefault(); // formulář nebude odeslán
});

var odkaz = document.querySelector("a");
odkaz.addEventListener("click", function(e) {
    e.stopPropagation(); // nikdo další se nedozví
});
```



DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Děkuji za pozornost

Martin Klíma