

Window, BOM a DOM

10.03.2022

Marek Nagy

window object

window

- hlavný objekt
- zodpovedá otvorenému oknu (záložky) prehliadača
- **globálne premenné a funkcie**
 - mapované ako vlastnosti tohoto objektu
 - pozor, aby ste si ich neprepísali
- napr. „rezervované“ vlastnosti
 - document, location, screen, history, navigator, ...
- funkčnosť závisí aj od konfigurácie prehliadača

Rozmery okna

- čistá (vnútorná) veľkosť okna (v pixeloch)

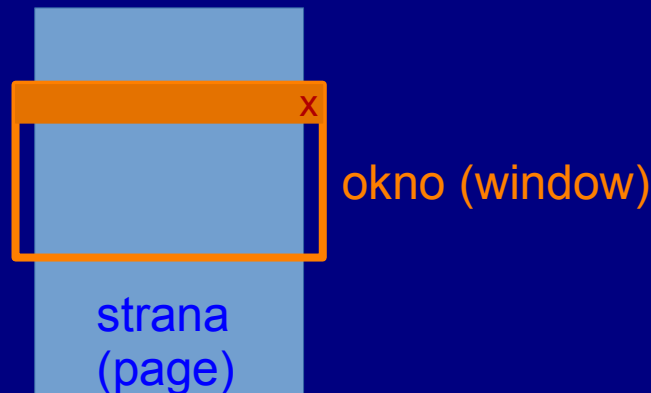
```
console.log (window.innerHeight, window.innerWidth);
```

- celá veľkosť okna aj s dekoráciou (v pixeloch)

```
console.log (window.outerHeight, window.outerWidth);
```

- koľko je strana vyskrolovaná (v pixeloch)

```
console.log (window.pageXOffset, window.pageYOffset);
```



Manipulácia s oknom

- fokus a zrušenie fokusu

```
window.focus();  
window.blur();
```

- vyskroluje stranu v okne (v pixeloch)

```
window.scrollTo (x, y);  
window.scrollBy (dx, dy);
```

- zmena rozmerov okna (v pixeloch) 

```
window.resizeTo (width, height);  
window.resizeBy (dWidth, dHeight);
```

Uvedené metódy nie vždy fungujú. Závisí od konfigurácie prehliadača.

- zmena pozície okna, (0,0) → vľavo hore

```
window.moveTo (x, y);  
window.moveBy (dx, dy);
```

Globálne metódy

- časovače

```
window.setInterval();  
window.clearInterval();  
window.setTimeout();  
window.clearTimeout();
```

Uvádzanie objektu **window** nie je potrebné. Defaultne sa tam hľadajú globálne metódy.

- konverzia binary ↔ base64 (ascii)

```
window.btoa ("Ahoj"); // "QWhvag=="  
window.atob ("Qwhvag=="); // "Ahoj"
```

- otvorenie nového okna

```
const win = window.open ("https://www.mmcitanka.sk");
```

- zatvorenie okna

```
window.close (); // Aktuálne okno  
win.close (); // Určené okno
```

Otvorí sa zväčša ako záložka prehliadača.

Media queries

Media query string.
Podmienky na vlastnosti zariadenia.

- aktuálne info o zariadení (podobne ako **@media** v CSS)

```
if (window.matchMedia("(max-width: 700px)").matches) {  
    /* šírka okna je <=700 px */  
} else {  
    /* šírka je >700 px */  
}
```

- zavesenie callback funkcie
 - volaná pri zmene → responsive design


```
function check (e) { console.log (e.matches) }  
  
let mm = window.matchMedia("(min-width: 700px)");  
  
check (mm); // Zavolá sa check() na začiatku  
  
mm.addEventListener ('change', check); // Volá sa pri zmene
```


BOM

Browser Object Model

navigator

- info o prehliadači (zariadení)
- napr. vlastnosti:
 - **onLine** - či je prehliadač on-line



V budúcnosti tu bude aj
info o pamäti, počte
CPU jadier, sieťových
parametroch uplink,
downlink, ...

```
console.log("Browser je "+ navigator.onLine? 'online' : 'offline');
```


- **geolocation** - GPS súradnice
- **userAgent** - identifikácia prehliadača

```
console.log (navigator.userAgent);  
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like  
Gecko) Chrome/65.0.3325.162 Safari/537.36
```

- **maxTouchPoints** – počet možných dotykov

screen

- info o obrazovke-monitore
- napr. vlastnosti:
 - **availHeight, availWidth**
 - rozmery bez dekorácií (v pixeloch)
 - **height, width**
 - celkové rozmery (v pixeloch)
 - **orientation**
 - natočenie, uhol natočenia



Možno využiť
napr. pri hrách
s tabletom.

history

- história navštívených stránok
- nie je možné zistiť konkrétne URL z histórie
- možnosť dať zobrazit' stránky z histórie

```
window.history.length
```

```
history.back ();           // Späť  
history.forward ();        // Dopredu
```

```
history.go (-1);           // späť o jeden krok  
history.go (3);            // dopredu o tri kroky
```

location

- čítanie aj manipulácia s aktuálne otvorenou URL
- napr. vlastnosti:
 - href → (celá URL)

hostname port pathname search

https://www.moja.com:9008/tajne/index.html?id=3345345&sec=4

origin

```
console.log (window.location.origin);  
https://www.moja.com:98
```


DOM

Document Object Model

document

- reprezentácia HTML dokumentu
 - strom s vrcholmi
 - document je zároveň aj najvrchnejší vrchol stromu
- vlastnosti:
 - title**, **head**, **scripts** - zodpovedá elementom `<title>`, `<head>`, `<script>`
 - body** - zodpovedá elementu `<body>`

```
window.document.body.innerHTML = "Ahoj stránka";
```


Príklad DOM

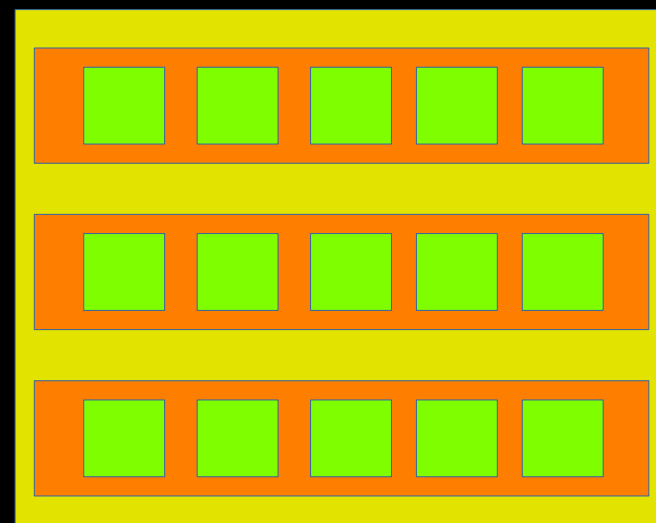
```
<body>
  <div class='Board'>

    <div class='Row'>
      <div class='Cell'></div>
      <div class='Cell'></div>
      <div class='Cell'></div>
    </div>

    <div class='Row'>
    </div>

    <div class='Row'>
    </div>

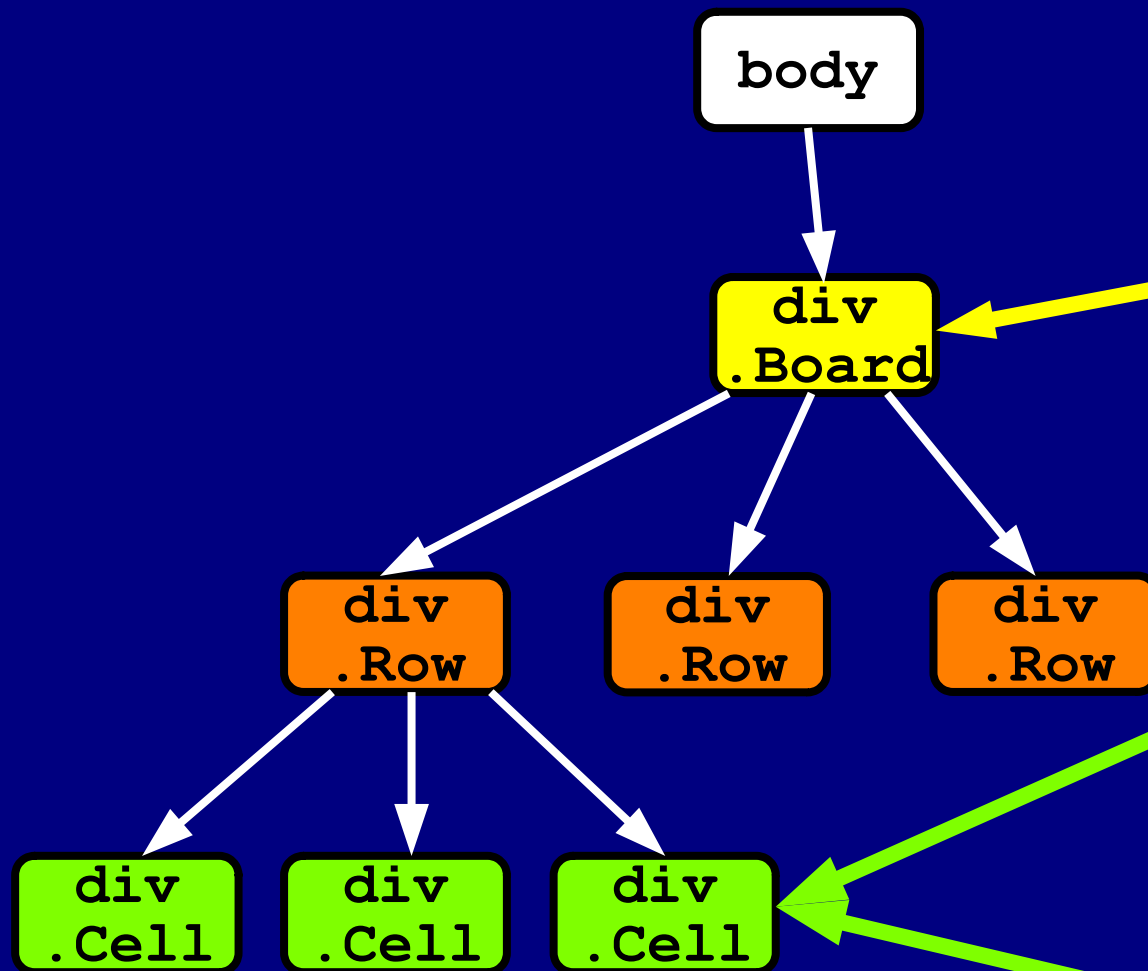
  </div>
</body>
```



Reprezentácia
štvorcovej
hracej plochy.

DOM

JS



```
class Board {  
  constructor () {  
    this.root  
    this.cells[i][j] = new Cell()  
  }  
}
```

```
b = new Board ()
```

```
b.cells[0][2].root
```

```
class Cell {  
  constructor () {  
    this.root  
  }  
}
```


Trieda Node

- objekty reprezentujú vrcholy DOM stromu
- HTML elementy, atribúty, texty, komentáre

Vytvorenie uzlov

- nový uzol (element) podľa HTML značky

```
nd = document.createElement ('div');
```

- uzol (komentár)

```
nd = document.createComment ('Poznámka v HTML');
```

- špeciálny uzol ako atribút HTML elementu

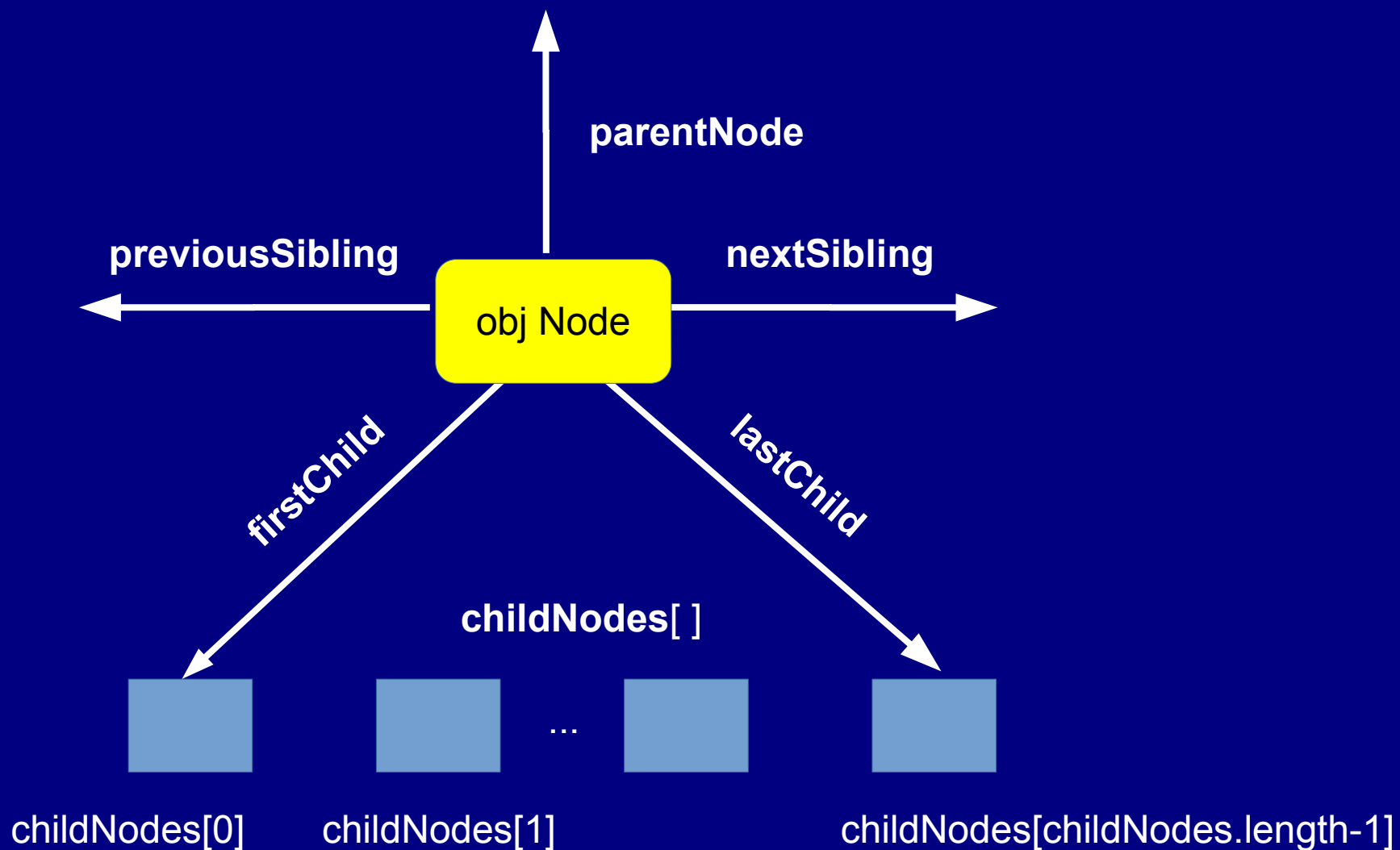
```
nd = document.createAttribute ('src');
```

- klonovaním

```
nd1 = nd.cloneNode ();           // klonuje iba daný uzol
```

```
nd2 = nd.cloneNode (true);       // Klonuje celý podstrom
```

Okolie uzla



Node

- pridať za posledné dieťa

```
e1.appendChild (e12);
```

– ak už uzol mal otca, „prevesí sa“

- odstrániť dieťa

```
e1.removeChild (e12);
```

- pridať pred dané dieťa

```
e1.insertBefore (e12, e1.childNodes[0] );
```

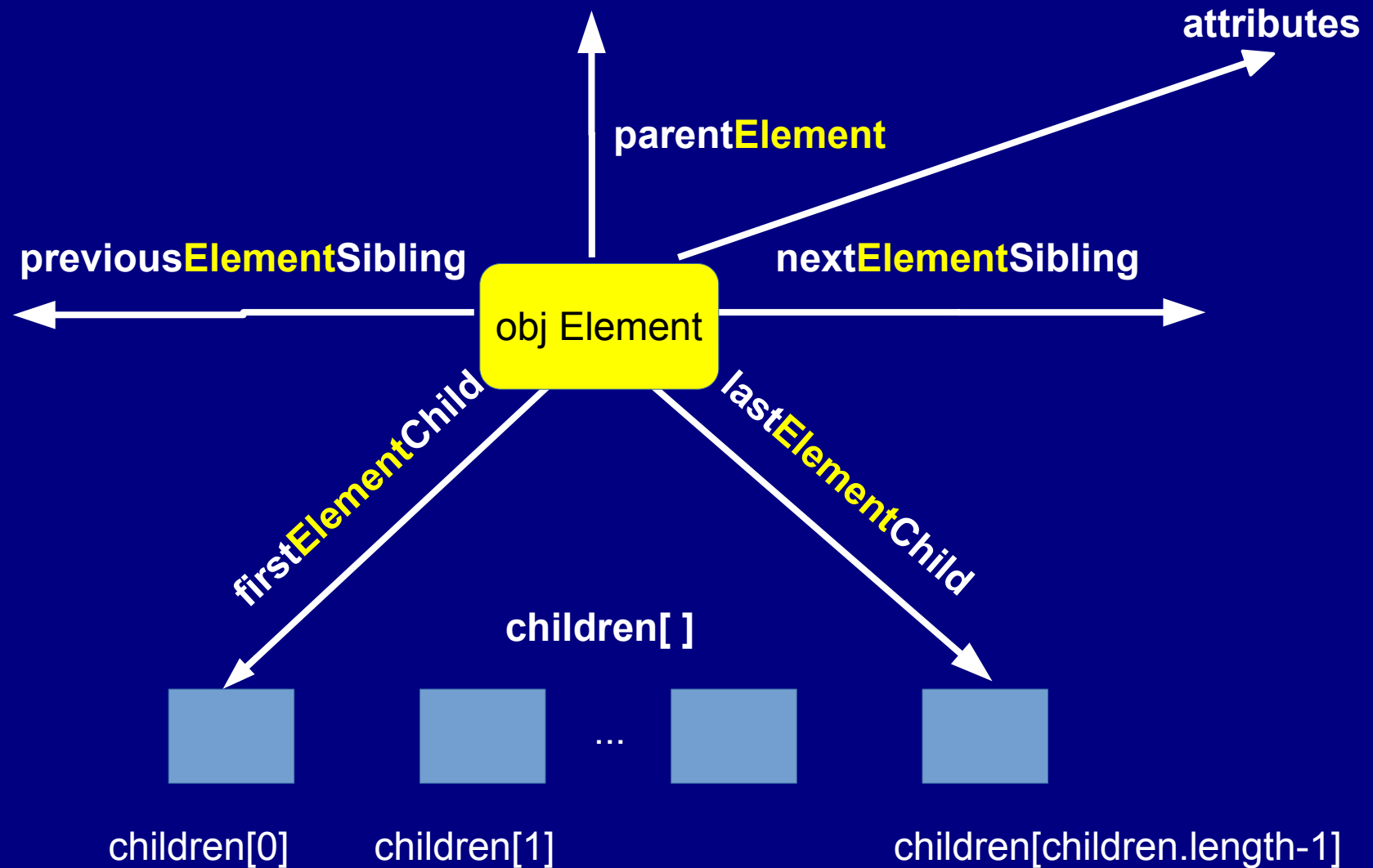
- vymeniť dieťa

```
e1.replaceChild (e12, e1.childNodes[7] );
```


Trieda **Element**

- špeciálny typ uzlov
- objekty reprezentujúce HTML elementy
- vytvárajú mapovanie HTML štruktúry dokumentu

HTML okolie elementu



Element

- pridať za posledné dieťa

```
e1.append (e12);
```

- pridať pred prvé dieťa

```
e1.prepend (e12);
```

- zameniť všetky deti

```
e1.replaceChildren (e12);
```

Element

- pridať za element ako súrodenca

```
e1.after (e12);
```

- pridať pred element ako súrodenca

```
e1.before (e12);
```

- odstrániť element

```
e1.remove ();
```

- vymeniť element

```
e1.replaceWith (e12);
```

Obsah elementu

- priamo zadany HTML reťazec

```
el.innerHTML = "<b>Ahoj</b>";  
  
el.innerHTML += '<span>Auto</span>';  
// Najprv sa vygeneruje HTML reťazec, zmení sa a znova sa  
parsuje!!!
```

– reťazec sa parsuje !

- textový obsah elementov

```
console.log (el.textContent); // Vyzbierajú sa všetky texty detí  
el.textContent = "Ahoj";
```


Vyhľadávanie elementov v podstróme

- podľa id

```
let e15 = e1.getElementById ("5622");
```

- podľa mena CSS triedy

```
let e13 = e1.getElementsByClassName ("moja");
```

- vyhľadá elementy daného typu

```
let e13 = e1.getElementsByTagName ("div");
```

- zadávanie CSS selektora

```
let e13 = e1.querySelector ("div.moja");
```

```
let elems = e1.querySelectorAll ("div.moja");
```


Funkcie \$, \$\$

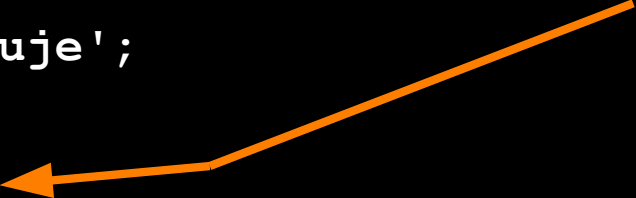
- konvencia
- treba si ich dodefinovať

```
const $ = qs => document.querySelector (qs);  
const $$ = qs => document.querySelectorAll (qs);
```

```
$('div').innerHTML = 'Funguje';
```

```
zs = $$('.pom');
```

```
for (let z of zs) z.innerHTML = 'Haha';  
// for (i = 0; i < zs.length; i++) zs[i].innerHTML = i;
```

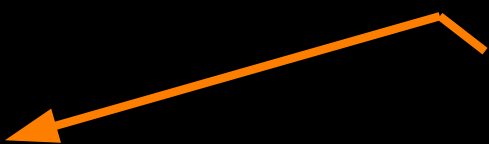


Nevráti klasické
pole, ale
iterovateľnú **kolekciu**
vyhovujúcich uzlov.

HTML atribúty elementu

- pridané metódy `setAttribute`, `getAttribute`, `hasAttribute`, `removeAttribute`
- elementy zvyknú mať namapované niektoré atribúty ako vlastnosti objektu
 - treba pozrieť manuál, ktoré to sú

```
let el = document.createElement ('img');  
el.setAttribute ('src', 'moj.jpg');  
if (el.hasAttribute ('src')) rElem.appendChild (el);  
if (el.getAttribute ('src') == 'ahoj.jpg') ;  
el.removeAttribute ('src');  
  
el.src = 'novy.jpg'  
console.log (el.width, el.height)
```



Ak je atribút
namapovaný
(cez setter/getter)
ľahšie sa s ním
manipuluje.

CSS elementu

- **style** – HTML atribút **style**
 - nenastavovať cez `setAttribute` !!!

```
document.body.style.backgroundColor = "red";  
el.style.fontSize = '20px';
```

Možno si za tým predstaviť getter/setter metódy, ktoré parsujú hodnotu atribútu **style**.

- **classList** – HTML atribút **class**
 - nenastavovať cez `setAttribute` !!!

```
document.body.classList.add ('mojaTrieda', 'vysviet');  
el.classList.remove ('moja', 'tvoja', 'nasa');  
el.classList.toggle ('moja');  
if (el.classList.contains ('moja')) el.innerHTML = 'Hura';
```


Zobrazenie elementu

- fokus a zrušenie fokusu

```
elem.focus(); elem.blur();
```

- koľko je vyskrolovaný element

```
console.log (elem.scrollTop, elem.scrollLeft);  
elem.scrollTop = 50;
```

- viditeľná veľkosť elementu + padding

```
console.log (elem.clientHeight, elem.clientWidth);
```

- viditeľná veľkosť + padding+border+scrollbar, (bez marginu)

```
console.log (elem.offsetHeight, elem.offsetWidth);
```

- veľkosť elementu aj so zaskrolovaním + padding

```
console.log (elem.scrollHeight, elem.scrollWidth);
```

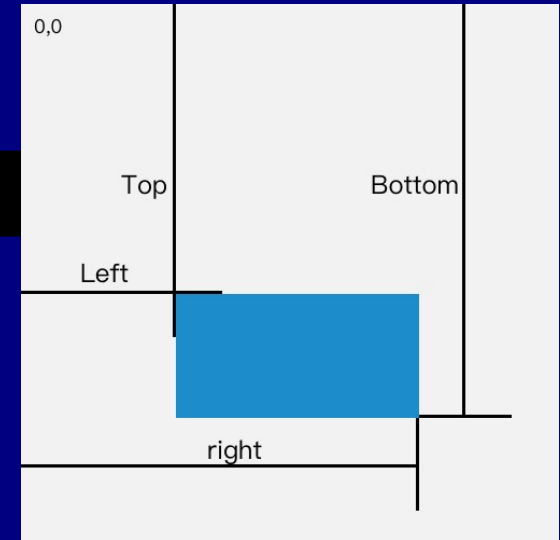
Rozmery elementu

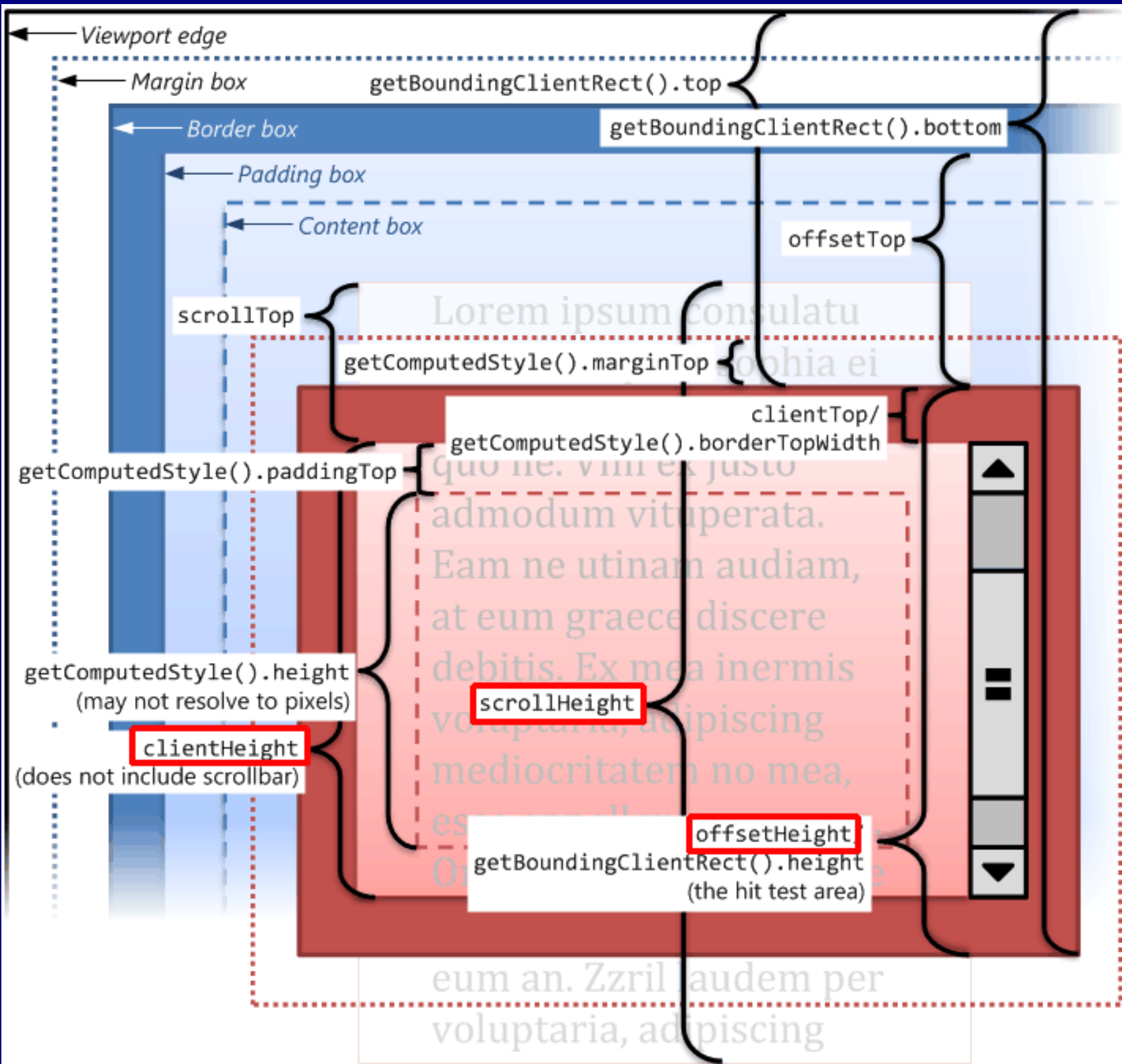
- kolekcia boxov detí

```
elem.getClientRects()
```

- jednotiaci box (obálka)

```
let {x,y, left,top, right,bottom, width,height} =  
    elem.getBoundingClientRect()
```





Udalosti

- **myšacie**
 - mousedown, mouseup, mousemove, mouseenter, mouseleave
- **klávesnicové**
 - keydown, keyup
- **dotykové**
 - touchstart, touchmove, touchend

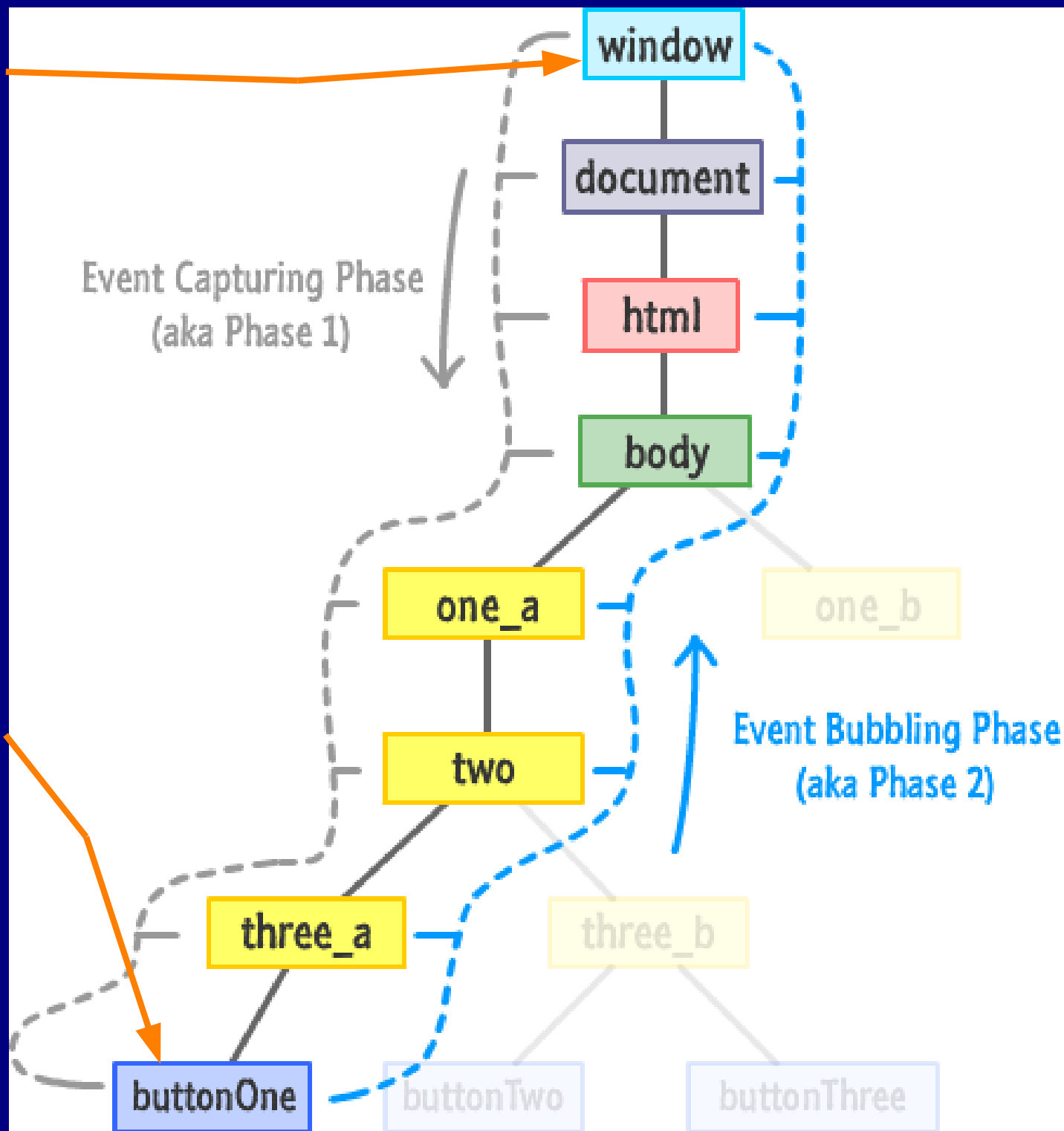
Šírenie udalosti

- stanovenie **target** (cieľového) element-u
 - napr. kam sa kliklo, dotklo, ...
 - napr. kde je focus pri klávesnici, ...
- najprv **capturing** fáza
 - od koreňa dokumentu ku cieľu
- potom **bubbling** fáza
 - od cieľa ku koreňu dokumentu

Počiatkový
element. (Koreň
stromu.)

Event Capturing Phase
(aka Phase 1)

target element.
Zistí sa dopredu
napr. pri kliknutí.



Udalosti na elemente

- pripojenie callback funkcie

```
el.addEventListener("click", func); // bubbling fáza  
el.addEventListener("click", func, true); // capturing fáza
```

- rozlišuje sa, v ktorej fáze šírenia sa zavolá

- odpojenie

```
el.removeEventListener("click", func); // bubbling fáza  
el.removeEventListener("click", func, true); // capturing fáza
```

- rozlišuje sa medzi typmi (capturing/bubbling)

- listener dostane objekt s udalosťou

```
function func (e) {  
  console.log (e.type);  
}
```

Event objekt

```
e => { e.currentTarget } // element vykonávaného listeneru  
e => { e.target }        // cieľový element
```

- nevykoná sa defaultná obsluha

```
e => { e.preventDefault() }
```

- nespustia sa ďalšie listenery elementu
 - na udalosť elementu možno zavesiť viac listenerov

```
e => { e.stopImmediatePropagation() }
```

- udalosť sa ďalej nepropaguje

```
e => { e.stopPropagation() }
```

MouseEvent objekt

- navyše pridané súradnice
 - vzhľadom na veľkosť fyzickej obrazovky

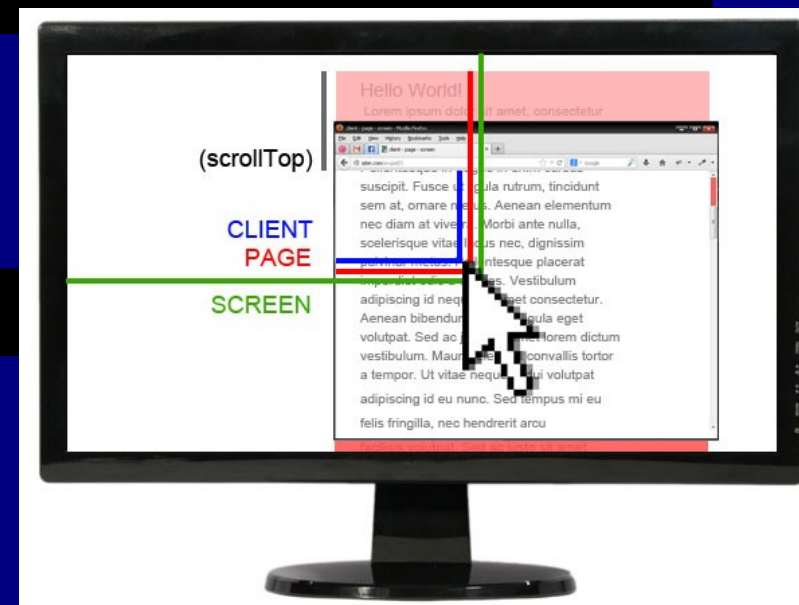
```
console.log (e.screenX, e.screenY);
```

- vzhľadom na zobrazovaciu oblasť prehliadača (skrolovanie sa ignoruje)

```
console.log (e.clientX, e.clientY);
```

- vzhľadom na celý dokument (môže byť vyskrolovaný)

```
console.log (e.pageX, e.pageY);
```



KeyboardEvent objekt

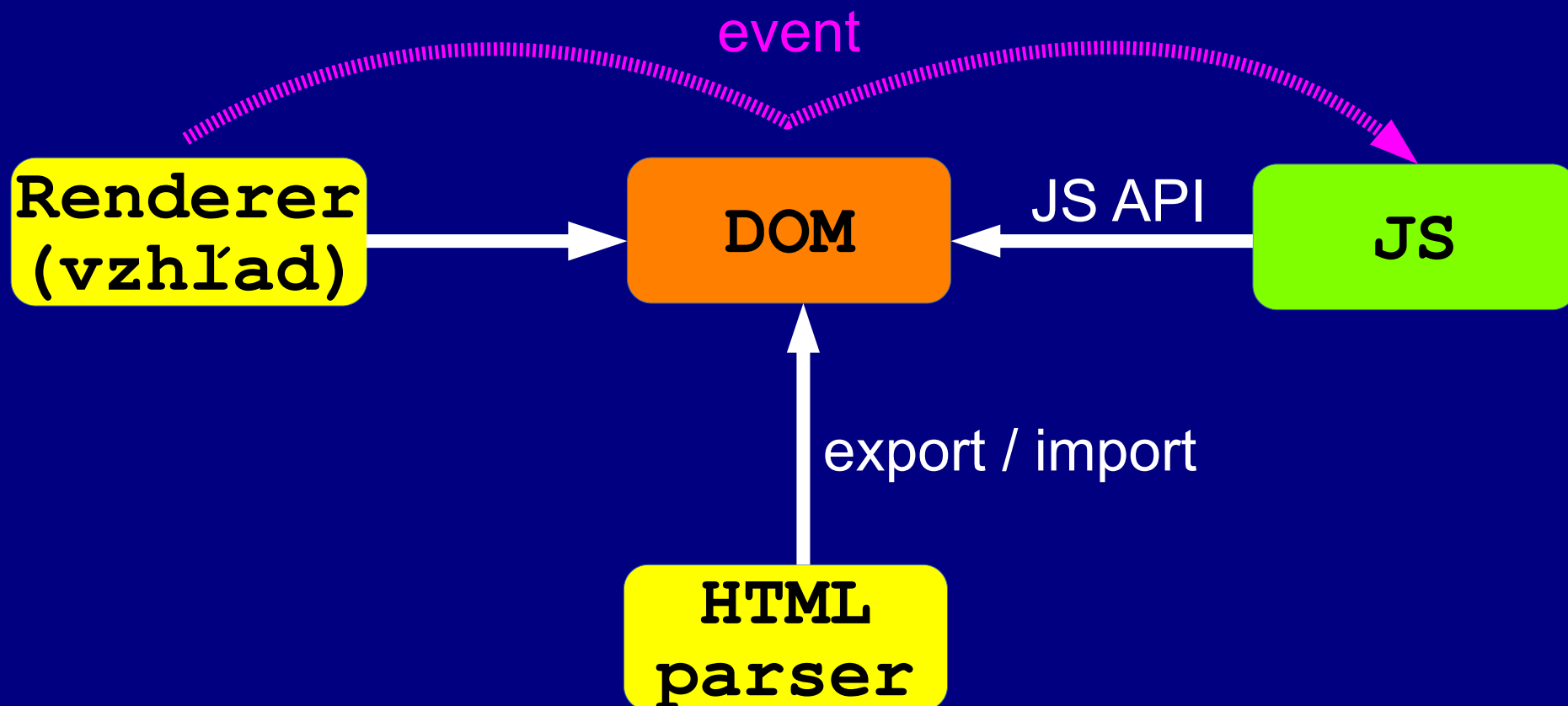
- info o stlačenej klávese
 - hodnota tlačidla klávesnice

```
console.log (e.key);
```

- označenie tlačidla klávesnice

```
console.log (e.code);
```


Architektúra



Ďakujem za pozornosť