



Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu

RAČUNARSKI PRAKTIKUM II

Predavanje 09 - HTML Document Object Model

28. travnja 2020.

Sastavio: Zvonimir Bujanović



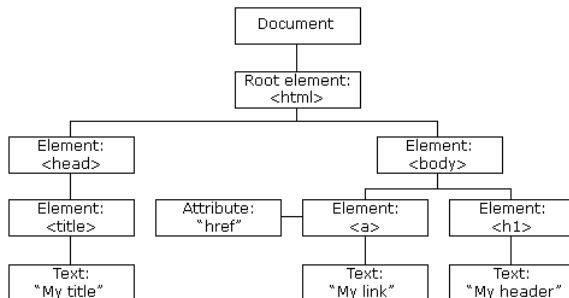
Document Object Model (DOM)

- API za HTML i XML dokumente. Neovisan o prog. jeziku, postoje implementacije u JavaScriptu, Pythonu, ...
- Reprezentira dokument u obliku stabla.
- Svaki čvor stabla je objekt koji ima svojstva, metode i eventualne funkcije kojima reagira na događaje.
- Omogućava programsko dodavanje, brisanje i modifikaciju pojedinih elemenata dokumenta – kako njegove strukture, tako i sadržaja i stila.
- Dobro je podržan u imalo novijim verzijama browsera (specijalno, IE 8- ima nestandardni API).

Ovdje dajemo samo letimičan pregled:

- referenca \rightsquigarrow [Mozilla Developer Network – DOM](#)
- u praksi se danas puno više koristi biblioteka [jQuery](#).

Reprezentacija HTML dokumenta pomoću DOM



```
1 <html>
2   <head>
3     <title>My title</title>
4   </head>
5   <body>
6     <a href="http://example.com">My link</a>
7     <h1>My header</h1>
8   </body>
9 </html>
```

JavaScript i DOM -- Primjer dinamičkog stvaranja elementa

```
1 <script>
2   window.onload = function() {
3     var h1 = document.createElement( "h1" );
4     var h1_text = document.createTextNode( "Naslov!" );
5     h1.appendChild( h1_text );
6     document.body.appendChild( h1 );
7   }
8 </script>
```

Napomena:

- Kako se učitava HTML dokument, tako se izvršava kod skripti.
- Često moramo pričekati dok se cijeli HTML dokument ne učitava prije manipuliranja njegovim elementima (ako je JavaScript kod iznad elemenata kojim manipulira, neće se ispravno izvršiti).
- U gornjem primjeru, kod funkcije se poziva nakon što se dogodi *dogadaj* `window.onload`, a to je upravo po učitavanju HTML dokumenta.

Node

- Bazna klasa kojom se reprezentiraju čvorovi u stablu DOM-a.
- Važnija svojstva:
 - `n.firstChild` – prvo dijete čvora `n` ili `null`.
 - `n.lastChild` – zadnje dijete čvora `n` ili `null`.
 - `n.nextSibling` – idući brat čvora `n` ili `null`.
 - `n.parentNode` – roditelj čvora `n` ili `null`.
 - `n.nodeName` – tip čvora `n` (na primjer "DIV").
 - `n.childNodes[k]` – k -to dijete čvora `n`.
 - `n.childNodes.length` – broj djece čvora `n`.
 - `n.children` – kao `n.childNodes`, ali ne uključuje komentare i text-čvorove.
- Važnije metode:
 - `n.appendChild(c)` – dodaje čvor `c` kao zadnje dijete čvora `n`.
 - `n.insertBefore(nc, c)` – dodaje dijete `nc` ispred djeteta `c` od `n`.
 - `n.removeChild(c)` – briše dijete `c` čvora `n`.
 - `n.replaceChild(nc, oc)` – zamjenjuje dijete `oc` čvorom `nc`.

Document

- Izvedena klasa iz `Node`, reprezentira korijen DOM stabla.
- Jedino dijete (`document.documentElement`) je čvor tipa `HTML`.
- Važnija svojstva:
 - `document.body` – referenca na body element (node)
 - `document.title` – naslov dokumenta (string)
 - `document.URL` – kompletna adresa dokumenta (string)
 - `document.referrer` – adresa stranice s koje je link doveo na ovu
 - `document.lastModified` – datum zadnje izmjene dokumenta
- Važnije metode:
 - `document.getElementById(id)` – vraća referencu na element sa zadanim id-om
 - `document.getElementsByTagName(t)` – vraća listu (ima `length`, `[]`) referenci na sve elemente sa zadanog tipa (`t="img", "div", "p"...`)
 - `document.getElementsByClassName(c)` – vraća listu (ima `length`, `[]`) referenci na sve elemente zadane klase `c`
 - `e=document.createElement(t)` – kreira novi element tipa `t`, ne dodaje ga još u stablo
 - `t=document.createTextNode(s)` – kreira čvor tipa `TextNode` u kojem se nalazi tekst stringa `s`. `TextNode` su listovi u DOM stablu.

Element / HTMLElement

- Izvedena klasa iz `Node`, reprezentira pojedini element dokumenta.
- Važnija svojstva:
 - `e.className` – daje klasu elementa `e` (kao string)
 - `e.id` – daje id atribut elementa `e` (kao string)
 - `e.innerHTML` – string koji sadrži HTML kod unutar elementa `e`
 - `e.clientHeight`, `e.clientWidth` – daje visinu (širinu) elementa u pixelima, uključuje padding ali ne border i margin.
- Važnije metode:
 - `e.getAttribute(a)` – vraća vrijednost (string) atributa `a` elementa `e`.
 - `e.removeAttribute(a)` – uklanja atribut `a` elementa `e`.
 - `e.setAttribute(a, s)` – postavlja string `s` kao vrijednost atributa `a` elementa `e`.

Element -- pristup CSS-u

Pomoću JavaScripta moguće je čitati i mijenjati stil elementa.

- `e.style.color` – boja teksta elementa `e`
- `e.style.fontFamily` – font-family svojstvo elementa `e`
- Općenito, CSS svojstvo oblika `ime-svojstva` prelazi u `e.style.imeSvojstva`.
- Čitanje svojstva na ovaj način detektira samo ona CSS svojstva eksplicitno postavljena u html dokumentu, a ne u vanjskoj CSS datoteci! Na primjer, ovako:

```
1 <p style="color: red;" id="p1">Crveno</p>
```

- Do svojstva kakvo je zaista prikazano u browseru dolazimo ovako:

```
1 var p = document.getElementById( "p1" );  
2 var style = document.defaultView.getComputedStyle( p, null );
```


Pojedini HTML elementi imaju izvedene klase koje olakšavaju manipuliranje specifičnim svojstvima. Na primjer:

- **HTMLTableElement** ima svojstva: `rows`, `tHead` i metode: `insertRow()`, `deleteRow()`.
- **HTMLTableRowElement** ima svojstva: `cells` i metode: `insertCell()`, `deleteCell()`.
- **HTMLImageElement** ima svojstva: `width`, `height`, `src`.

Primjer 1

Neka je zadan HTML kod kao u donjem primjeru (može biti po volji mnogo listi s po volji mnogo životinja).

Treba napisati skriptu koja će u konzolu ispisati tekst "Hello, *životinja*" za svaku od životinja, te ispisati ukupan broj životinja.

Velike životinje treba na web-stranici ispisati fontom od 24pt.

```
1 <html>
2   <head>...</head>
3   <body>
4     <ul>
5       <li>Pas</li>
6       <li>Mačka</li>
7       <li class="velika">Žirafa</li>
8     </ul>
9     <ul>
10      <li class="velika">Slon</li>
11      <li>Mrav</li>
12    </ul>
13  </body>
14 </html>
```

jQuery

- Biblioteka funkcija koja čini manipulaciju HTML dokumentima bitno jednostavnijim.
- Izuzetno popularna, praktički je postala standard.
- Podržava i događaje (events), animacije, Ajax.
- Postoje dva izdanja biblioteke:
 - `jquery.js` – nekomprimirana, podržava debugiranje;
 - `jquery.min.js` – komprimirana, za produkcijsku upotrebu.
- Dokumentacija:
 - [TutorialsPoint JQuery](#)
 - [w3schools JQuery](#)
 - [Quick API Reference](#)

```
1 <head>
2   <title>...</title>
3   <script src="https://ajax.googleapis.com/ajax/libs/
4                                   jquery/3.3.1/jquery.js"></script>
5 </head>
```

- Sva manipulacija HTML DOM-om obavlja se preko funkcije `$`.
- Alternativno, umjesto `$` može se koristiti `jQuery`.
- `$(sel)` vraća objekt tipa `jQuery` koji sadrži kolekciju svih HTML elemenata koji odgovaraju CSS selektoru `sel`.

```
1 // kolekcija svih ul elemenata iz HTML-a
2 var uls = $( "ul" );
3
4 // kolekcija svih elemenata klase veliki
5 var veliki = $( ".veliki" );
6
7 // kolekcija koja sadrži jedini element s id-om naslov
8 var naslov = $( "#naslov" );
9
10 // kolekcija svih paragrafa unutar sekcija klase clanak
11 // i svih naslova tipa h1
12 var para = $( "section.clanak p, h1" );
```

Neka svojstva i metode klase jQuery (neka je `j` objekt tipa jQuery):

- `j.length` – vraća broj elemenata u kolekciji;
- `j.eq(i)` – vraća jQuery objekt za $(i+1)$ -vi element u kolekciji (ako $i < 0$, onda broji od kraja);
- `j.css(prop)` – vraća vrijednost CSS svojstva `prop` za prvi element kolekcije;
- `j.css(prop, val)` – svim elementima kolekcije postavlja CSS svojstvo `prop` na vrijednost `val`;
- `j.html()` – vraća HTML sadržaj koji se nalazi unutar prvog elementa kolekcije;
- `j.html(x)` – svim elementima kolekcije postavlja HTML sadržaj na `x`;

```
1 // Primjer ulančavanja poziva funkcija u jQuery:  
2 // Treći h1 naslov će biti plave boje i imat će sadržaj "Yes!"  
3 $("h1").eq(2).css("color", "blue").html("Yes!");
```

Neka svojstva i metode klase jQuery (neka je `j` objekt tipa jQuery):

- `j.attr(a)` – vraća vrijednost atributa `a` za prvi element kolekcije;
- `j.attr(a, val)` – svim elementima kolekcije postavlja atribut `a` na vrijednost `val`;
- Postoji i `j.prop(a)` – razlika u detaljima.
- `j.val()` – vraća vrijednost (za elemente tipa `input`, `select`, `textarea`) za prvi element kolekcije;
- `j.val(x)` – svim elementima kolekcije postavlja vrijednost na `val`;

```
1 // Elementu (npr. input) s id-om ime postavlja vrijednost Mirko
2 $("#ime").val("Mirko");
3
4 // Svim select elementima postavlja vrijednosti jabuke i kruške
5 $("select").val("jabuke", "kruške");
```

Postoji još **puno funkcija**, npr:

- `$(html)` – stvara novi jQuery objekt sastavljen od zadanog html koda. Novi objekt nije uklopljen u stablo DOM-a.
- `j.append(x)`, `j.prepend(x)` – na kraj/početak HTML koda svih elemenata iz kolekcije se dodaje `x` (HTML ili jQuery objekt)
- `j.addClass(c)`, `j.removeClass(c)` – dodaje/uklanja CSS klasu `c` svim elementima kolekcije
- `j.children()` – kolekcija koja sadrži svu djecu svih elemenata iz `j`
- `j.next()` – kolekcija koja sadrži idućeg brata svih elemenata iz `j`
- `j.parent()` – kolekcija koja sadrži roditelje svih elemenata iz `j`
- `j.parents()` – kolekcija koja sadrži pretke svih elemenata iz `j`

```
1 var x = $( "<h1>Naslov!</h1>" ); // stvori novi element
2 $("body").prepend( x ); // stavi naslov na početak body-a
```

Zadatak 1

Na web-stranici se nalazi nekoliko paragrafa.

Sve paragrafe koji sadrže riječ "crveno", ispišite crvenom bojom.

Svim paragrafima koji sadrže riječ "žuto", postavite boju pozadine na žutu.

U jQuery umjesto `window.onload` koristimo:

```
1 $(document).ready( function() {  
2     // kod koji treba izvršiti kad se učitava web-stranica  
3 } );
```


Riješite **Primjer 1** koristeći jQuery.

Dodatno, procesirajte samo one ul-ove kojima je postavljena klasa `zivotinja`.

Stvorite novi element tipa `ul` na kraju web-stranice koji sadrži popis svih životinja u jednoj listi.

Dodajte uskličnik iza imena svake životinje u originalnim listama.

Browser Object Model

Document Object Model (DOM) \rightsquigarrow manipulacija HTML dokumentima.
Browser Object Model (BOM) \rightsquigarrow manipulacija browserom neovisno o prikazanom sadržaju.

BOM se sastoji od nekoliko objekata:

- **window** – glavni BOM objekt, (ne samo) za manipulaciju prozorom browsera.
- **window.location** – informacije o učitanoj dokumentu u browser. Uočiti: `window.location === document.location`, tj. `location` je svojstvo i `window` i `document` objekta.
- **window.navigator** – svojstva browsera, npr. verzija, operativni sustav, popis pluginova, jezik korisnikovog OS-a, ...
- **window.screen** – podaci o prozoru browsera, npr. njegova širina i visina, broj boja, ...
- **window.history** – povijest browsanja unutar aktualnog prozora (tj. taba)

Browser Object Model: window

window = "globalni objekt", svaka deklarirana varijabla je njegov član

```
1 function hello() { console.log( "Hello" ); }
2 var ime = "Pero";
3
4 console.log( ime ); // "Pero"
5 console.log( window.ime ); // "Pero"
6 window.hello(); // ispiše "Hello" u konzoli
```

Otvaranje novih prozora:

- `window.open()` – komplicirana sintaksa (vidi [Primjer 2](#))
- `alert(poruka)` – pop-up prozor u kojem piše poruka, klik na OK
- `confirm(poruka)` – pop-up prozor u kojem piše poruka, klik na OK ili Cancel, vraća true/false
- `prompt(poruka)` – pop-up prozor u kojem piše poruka, očekuje unos teksta i onda klik na OK ili Cancel, vraća utipkani string ili null

Dohvaćanje dimenzija prozora:

- `window.innerWidth`, `window.innerHeight`
- `document.documentElement.clientWidth` (IE8-)

Manipulacija dimenzijom i položajem prozora:

- Samo za prozore otvorene sa `window.open()`.
- `window.resizeTo(w, h)`
- `window.moveTo(x, y)`

Zatvaranje otvorenog prozora:

- `window.close()`

Primjer 2

Donji kod otvara www.google.hr u novom prozoru dimenzija 400x400, te taj prozor smjesti tako da mu je gornji lijevi kut na koordinatama (10, 10) ekrana.

Zatim promijeni dimenzije tog prozora na 700x400 i pomakne ga na druge koordinate. Na kraju u skočnom okviru ponudi korisniku da zatvori prozor.

Da bi primjer radio, treba u Firefoxu dozvoliti pop-up prozore za domenu na kojoj ga pokrećete.

```
1 var w=window.open(  
2     "http://www.google.hr",  
3     "goog",  
4     "height=400,width=400,top=10,left=10,resizable=yes" );  
5  
6 w.resizeTo( 700, 400 );  
7 w.moveTo( 50, 100 );  
8 if( confirm( "Zatvori prozor?" ) )  
9     w.close();
```

Rad s vremenskim intervalima (uoč: JavaScript je *single-threaded*!):

- `id=setTimeout(f, t)` – izvrši kod zadan funkcijom `f` nakon čekanja od `t` milisekundi
- `id=setInterval(f, t)` – izvršavaj kod zadan funkcijom `f` svakih `t` milisekundi
 - Nekad je bolje koristiti `setTimeout`, i onda ponovno u funkciji pozvati `setTimeout`.
 - Razlog je taj što samo izvršavanje funkcije `f` može trajati dulje od perioda `t`, pa `setInterval` ima nepredvidivo ponašanje.
- `clearTimeout(id)/clearInterval(id)` – otkaži štopericu `id`

Primjer 3

Sljedeći kod prikazuje štopericu unutar paragrafa s id-om p. Vrijeme u štoperici se osvježi svake sekunde.

```
1 <html>
2   <head></head>
3   <body>
4     <p id="p"></p>
5
6     <script>
7       function updateClock() {
8         var d = new Date();
9         $("#p").html( d.toLocaleTimeString() );
10      }
11
12      setInterval( updateClock, 1000 );
13    </script>
14  </body>
15 </html>
```

Zadatak 3

Postavite neku sliku na web-stranicu.

Zatim napravite sljedeće:

- Neka se slika postupno povećava do svoje dvostruke veličine.
- Nakon toga, slika se smanjuje do svoje originalne veličine.
- Ovaj postupak se ponavlja.

Napomene:

- `j.css("width")` vraća širinu s mjernom jedinicom, npr. "200px".
- `j.width()` vraća širinu bez mjerne jedinice, npr. "200".
- Umjesto `$(document).ready(...)` u ovom zadatku koristite `$(window).on("load", ...)`.
 - Prva funkcija se pozove čim se učitava cijeli HTML dokument i pripremi DOM stablo.
 - Druga se poziva kada se učitaju i sve slike i ostali eksterni elementi.

- Događaj (“event”) – omogućava izvršavanje JavaScript koda kada se pojavi određena akcija unutar browsera ili učitanoj dokumenta. Na primjer:
 - klik mišem na neki HTML element,
 - prelazak mišem preko nekog HTML elementa,
 - scroll sadržaja web-stranice,
 - utipkavanje teksta u neki `input` tag,
 - promjena veličine prozora browsera i slično.
- Pojedine funkcije se mogu “pretplatiti” na događaje:
 - Čim se neki događaj pojavi, automatski se pozivaju sve funkcije pretplaćene na njega.
 - Ovaj model se tradicionalno naziva *Observer pattern*.

Događaji unutar HTML-a

Svaki događaj na kojeg pojedini HTML element reagira može imati pridruženu akciju preko odgovarajućeg HTML atributa.

```
1 <!-- Uoči jednostruke navodnike oko parametra od alert!  
2      this = element koji je generirao događaj (input). -->  
3 <input type="button" value="Klikni me"  
4       onclick="alert('Na gumbu pise ' + this.value);" />  
5  
6 <script>  
7     function handler() { alert( "Klik na button!" ); }  
8 </script>  
9 <input type="button" value="Klikni" onclick="handler();" />
```

Ovaj pristup se izbjegava jer ima nekoliko mana:

- Dogodi se greška ako se događaj dogodi prije no što se učitava kod funkcije koja na njega reagira.
- Ne poštuje se ideja razdvajanja sadržaja dokumenta (HTML) i dinamičkog aspekta dokumenta (JavaScript).

Prva alternativa: DOM Level 0 Event Handler

Funkcija koja reagira na događaj se dodaje kao član odgovarajućeg čvora u DOM stablu.

```
1 <input type="button" id="btn1" value="Klikni me!" />
2
3 <script>
4     var btn = document.getElementById( "btn1" );
5     btn.onclick = function() {
6         alert( "Klik na gumb na kojem piše " + this.value );
7     };
8 </script>
```

Funkcija koja reagira na događaj ("event handler") se uklanja ovako:

```
1 btn.onclick = null;
```

Ovaj pristup funkcionira u svim browserima.

Druga alternativa: DOM Level 2 Event Handler

Funkcija koja reagira na događaj se “pretplati” na odgovarajući događaj.

```
1 <input type="button" id="btn1" value="Klikni me!" />
2
3 <script>
4     var btn = document.getElementById( "btn1" );
5     btn.addEventListener( "click", function() {
6         alert( "Klik na gumb " + this.value );
7     }, false );
8
9     var f = function() { alert( "Još jedan prozor!" ); };
10    btn.addEventListener( "click", f, false );
11 </script>
```

Funkcija koja reagira na događaj (“event handler”) se uklanja ovako:

```
1 btn.removeEventHandler( "click", f, false );
```

Treća alternativa: jQuery (Primjer 4)

jQuery objekt se pretplati na neki događaj ovako:

```
1 <input type="button" id="btn1" value="Klikni me!" />
2
3 <script>
4     var btn = $( "#btn1" );
5     btn.on( "click", function() {
6         // $(this) napravi jQuery objekt od HTML elementa
7         // this na kojem je nastao event
8         alert( "Klik na gumb " + $(this).val() );
9     } );
10
11     var f = function() { alert( "Još jedan prozor!" ); };
12     btn.on( "click", f );
13 </script>
```

Funkcija koja reagira na događaj ("event handler") se uklanja ovako:

```
1 btn.off( "click", f );
```

Zadatak 4

Napišite JavaScript program koji simulira kalkulator.

Jednostavna varijanta:

<input type="text" value="10"/>	<input type="text" value="20"/>	<input type="button" value="+"/>	<input type="button" value="-"/>	<input type="button" value="*"/>	<input style="border: 2px solid blue;" type="button" value="/"/>	<input type="text" value="0.5"/>
---------------------------------	---------------------------------	----------------------------------	----------------------------------	----------------------------------	--	----------------------------------

Komplicirana varijanta (izazov: što kraći program!)

<input type="text" value="0"/>			
7	8	9	+
4	5	6	-
1	2	3	*
0	=	C	/

- Događaji korisničkog sučelja (**UIEvent**)
 - **load** – kada se odgovarajući element u potpunosti učitava. Najčešće se koristi za **window**: opali se kada se cijela web-stranica, zajedno sa slikama i ostalim elementima učitava.
 - **unload** – kada prelazimo s jedne stranice na drugu, za oslobađanje memorije i sprečavanje memory leak-ova.
 - **resize** – kada se promijeni veličina prozora browsera ili se on minimizira/maksimizira.
 - **scroll** – kada se pomiče prikazani sadržaj prozora. Trenutni odmak od vrha prozora možemo dobiti pomoću `document.body.scrollTop`.

Napomena:

- Za događaj *event*, odgovarajući HTML atribut ili DOM 0 svojstvo se zove *onevent*: `load` \rightsquigarrow `onload`, `scroll` \rightsquigarrow `onscroll`, ...

- Događaji fokusa (**FocusEvent**)
 - **focus** – kada element dobije fokus (npr. klikom miša na `input` možemo unositi tekst)
 - **blur** – kada element izgubi fokus (jer npr. kliknemo mišem na neki drugi)
- Događaji s mišem (**MouseEvent**)
 - **click**, **dblclick** – klik/dupli klik na lijevu tipku miša
 - **mousedown** – klik bilo koje tipke miša
 - **mouseup** – otpuštanje bilo koje tipke miša
 - **mouseenter**, **mouseleave** – ulazak/izlazak miša u područje koje zauzima neki element
 - **mousemove** – pomicanje miša iznad elementa

Vrste događaja

- Događaji s tipkovnicom (**KeyboardEvent**)
 - **keydown** – pritisnuta je tipka (događaj se ponavlja sve dok je pritisnuta)
 - **keyup** – tipka je otpuštena
- Događaj **input** – korisnik nešto unosi u textbox/textarea ili je kliknuo mišem drugi element tipa input.
- Postoje još razni drugi događaji, npr. **contextmenu** za desni klik miša, **orientationchange** za promjenu orijentacije mobilnog uređaja itd. Vidi [ovdje](#).

Funkcije koje reagiraju na događaj mogu primiti jedan parametar. Taj parametar će se prilikom poziva funkcije automatski popuniti informacijama o događaju:

- Događaji s mišem će (na primjer) dati koordinate klika.
- Događaji s tipkovnicom će (na primjer) dati podatke o tome koja tipka je pritisnuta i je li pri tome bio pritisnut shift/alt/control.

Zadatak 5

Napravite web-stranicu na kojoj se nalaze text-box i gumb. U text-box korisnik treba unijeti datum u formatu `dd/mm/yyyy`.

Ako je u text-boxu datum ispravan, gumb treba biti zelen, na njemu treba pisati "Datum je OK!" i na njega se može kliknuti.

Ako datum u text-boxu nije ispravan, gumb treba biti crven, na njemu treba pisati "Datum nije OK!" i na njega se ne može kliknuti.

Uputa:

- Textbox treba reagirati na događaj `input`.
- Ako atribut `disabled` ne postoji ili je prazan string, onda je gumb klikabilan.

HTML element div (plavi pravokutnik) slijedi pokrete miša.

Ako uz pokrete miša držimo tipku `r`, pravokutnik je crven.

Ako uz pokrete miša držimo neku drugu tipku, pravokutnik je zelen.

Zadatak 6

Napravite web-stranicu koja ispisuje koordinate na kojima se nalazi miš (događaj `mousemove` na elementu `body`).

Kada korisnik napravi klik, treba ispisati kojom tipkom miša je napravio klik (događaj `mousedown`; `event.button` daje tipku).

Na stranici se nalazi i jedan gumb. Kada korisnik pokuša doći mišem iznad gumba (događaj `mouseover`), gumb treba "preseliti" na neko drugo, slučajno odabrano mjesto.

Upute:

- `html{ height: 100%; }, body{ min-height: 100%; }`
- Za gumb stavite CSS svojstvo `position: absolute`; te mu udaljenost od lijevog ruba postavljajte pomoću CSS svojstva `left`, a od gornjeg pomoću CSS svojstva `top`.
- `Math.floor(Math.random() * 1000)` – slučajan cijeli broj između 0 i 999.

Događaji za dinamički kreirane elemente

Za element koji kreiramo pomoću JavaScripta (a ne iz HTML-a) kažemo da je kreiran dinamički.

Ako takvom elementu nakon kreiranja dodamo *event handler*, to radi kako je i očekivano.

```
1 $( document ).ready( function()
2 {
3     var btn =
4         $( "<button>Klikni me!</button>" )
5         .addClass( "klasa" )
6         .on( "click",
7             function() { alert( "Klik!" ); } );
8
9     $( "body" ).append( btn );
10    // btn sad reagira na klik.
11 } );
```

Događaji za dinamički kreirane elemente

Međutim, ako unaprijed definiramo *event handler*, pa onda kreiramo element, on neće reagirati na događaj!

```
1 $( document ).ready( function()  
2 {  
3     $( "button.klasa" )  
4         .on( "click",  
5             function() { alert( "Klik!" ); } );  
6  
7     var btn =  
8         $( "<button>Klikni me!</button>" )  
9         .addClass( "klasa" )  
10  
11     $( "body" ).append( btn );  
12     // btn NE REAGIRA na događaj!!!  
13 } );
```

Događaji za dinamički kreirane elemente

Za ovo ipak postoji rješenje: *event handler* treba dodati **roditelju** elementa kojeg ćemo kasnije stvoriti (ili **body**-ju).

Kažemo da je takav događaj *delegiran*.

```
1 $( document ).ready( function()
2 {
3     // Drugi parameter ("button.klasa") je CSS selektor
4     // podelemenata od body.
5     $( "body" ).on( "click", "button.klasa",
6         function() { alert( "Klik!" ); } );
7
8     var btn =
9         $( "<button>Klikni me!</button>" )
10         .addClass( "klasa" );
11
12     $( "body" ).append( btn );
13     // btn sad REAGIRA na događaj!!!
14 } );
```