Osnove web programiranja

JavaScript biblioteke i jQuery

Termin 10 i 11

Sadržaj

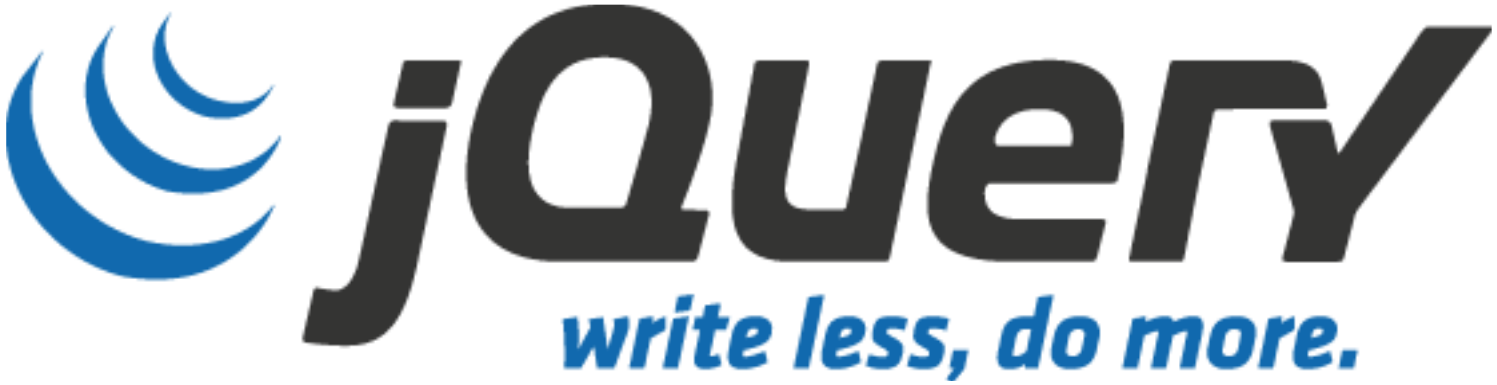
1. *JavaScript* biblioteke
2. *jQuery*

JavaScript biblioteke

- *JavaScript* biblioteka obuhvata skup konstanti, klasa i funkcija koje su definisane u eksternoj *.js* datoteci
- *JavaScript* biblioteka obuhvata jednu ili više eksternih *.js* datoteka koje se uključuju na HTML stranicu pre *JavaScript* skripte koja će je koristiti

```
<script src="javascript_bibilioteka.js" type="text/javascript"></script>
<script src="js/skripta_koja_koristi_biblioteku.js" type="text/javascript"></script>
```

jQuery



JavaScript biblioteka kojom se pojednostavljuje programiranje na klijentskoj stani tj. pojedostavljuje se:

- pristup elementima *web* stranice
- izmena izgleda *web* stranice
- izmena sadržaja *web* stranice
- interakcija sa korisnikom
- animacije
- uobičajene *JavaScript* naredbe
- Dobavljanje sadržaja sa servera bez ponovnog osvežavanja stranice (AJAX zahtevi i odgovori)

jQuery

Kada bi u Google ukucali pretragu jQuery (decembar 2020)

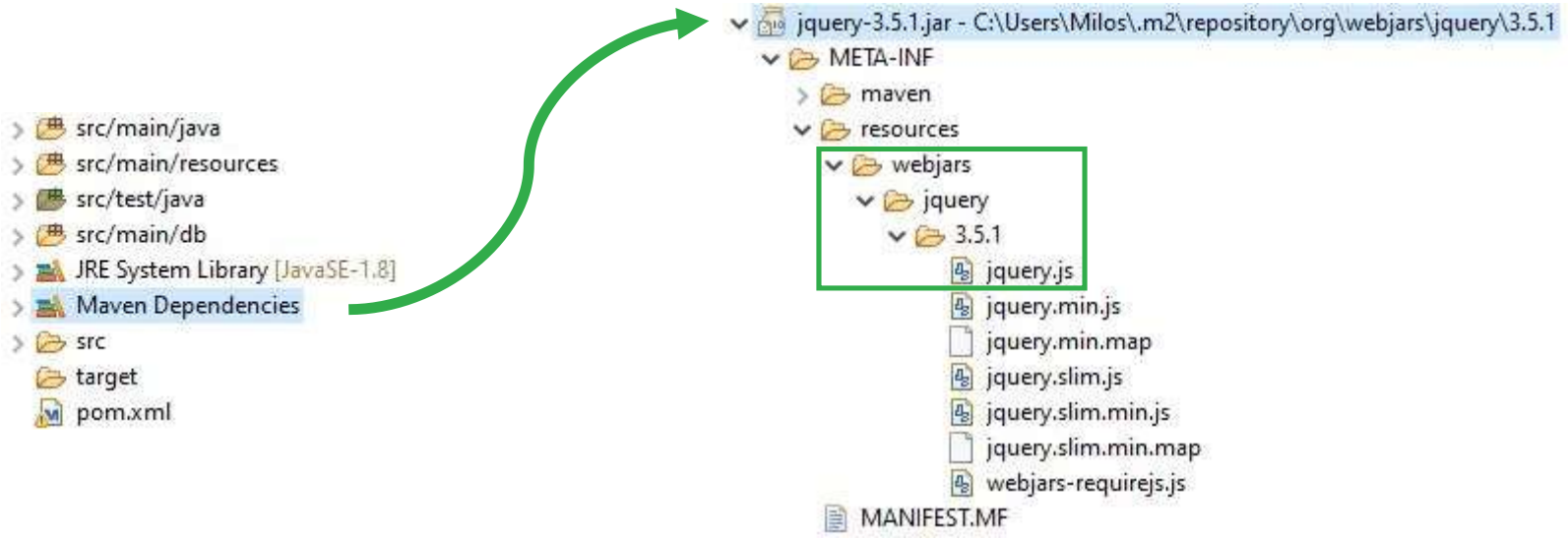
Podešavanje

- da bi se *jQuery* biblioteka uključila u *Maven* projekat, sledeća međuzavisnost se mora dodati u *pom.xml* datoteku:

```
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>jquery</artifactId>
    <version>3.5.1</version>
</dependency>
```

Podešavanje

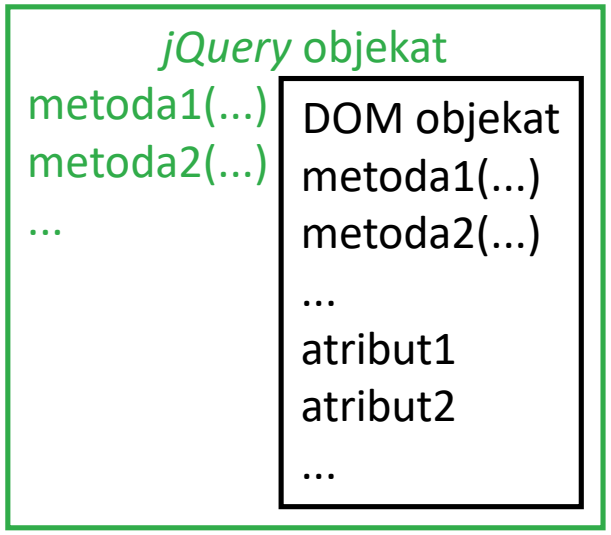
- da bi se *jQuery* biblioteka uvezla na HTML stranicu, sledeći *script* element mora biti definisan na HTML stranici pre *script* elementa u kome se koriste *jQuery* pozivi:

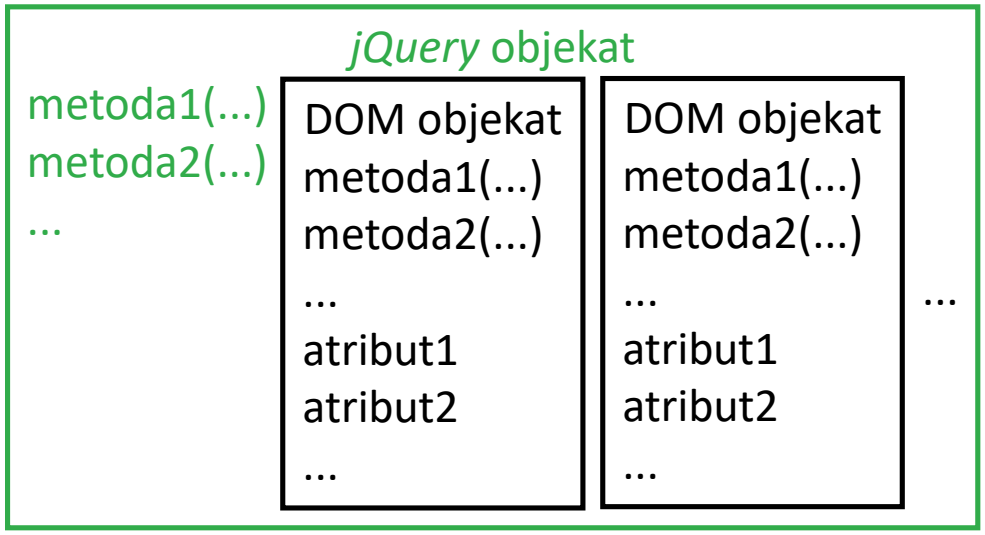



```
<script src="webjars/jquery/3.5.1/jquery.js" type="text/javascript"></script>
<script src="js/skripta_koja_koristi_jquery.js" type="text/javascript"></script>
```

Ideja

- DOM objekat ili skup objekata se zatvaraju (*wrap*-uju) u *jQuery* objekat i njima se rukuje uz pomoć metoda *jQuery* objekta
- ove metode tipično nude lakše rukovanje DOM objektima i dodatno nude složenije operacije koje DOM objekti ne implementiraju po specifikaciji
- jQuery objektima se rukuje gotovo univerzalno putem metoda





Sintaksa

- DOM objekat do čije se reference došlo klasičnim putem preko *JavaScript*-a može se zatvoriti (*wrap*-ovati) u *jQuery* objekat i sačuvati u posebnu referencu
- od tog momenta nad tom referencom se mogu pozvati *jQuery* metode
- ako je potreban poziv samo jedne metode, prethodno se može zapisati u jednom izrazu
- **$** : koristi jQuery - aktivira jQuery

```
var domObjekat = document.getElementById("id")

…

var jQueryObjekat = $(domObjekat)

…

jQueryObjekat.jQueryMetoda(…)
```

```
var domObjekat = …

…

$(domObjekat).jQueryMetoda(…)
```

Sintaksa

- ukoliko DOM objekti nisu na raspolaganju, *jQuery* biblioteka može da obavi efikasnu pretragu i nađe ih po CSS selektoru
- Jednostavno **selektovanje** HTML elemenata i **izvršavanje akcija** nad njima **$(slektor).akcija()**
- **$** : koristi jQuery - aktivira jQuery
- **(selektor)** : selektuj HTML element, kao u CSS
- **akcija()** : izvrši akciju nad selektovanim elementom

- ovaj pristup vraća već *wrap*-ovane DOM objekte kao *jQuery* objekat

```
var jQueryObjekat = $("CSS selektor")
```

Aktiviranje

- Sve počinje

```
$(document).ready(function() {
   ...
});
```

funkcijom, unutar <script> taga u <body> sekciji.

- DOM stablom ne može da se manipuliše ispravno sve dok se on ne prevede u stanje “ready”. JQuery detektuje stanje spremnosti.
- **$(document).ready(function() { ... })** – kod uključen u okviru funkcije će se izvršiti samo kada je DOM stablo spremno za JavaScript kod da se izvrši.
- Ovim se izbegava da se jQuery kod izvršava pre nego što se završi učitavanje stranice.

Aktiviranje

primer1.html

- **$(window).on("load", function() { ... })**– kod uključen u okviru funkcije će se izvršiti samo kada je cela stranica (npr. slike ili iframes), a ne samo DOM stablo, spremna za JavaScript kod da se izvrši.
- Primer:

```
$(this).hide(); // sakriva trenutni element
$("p").hide(); // sakriva sve <p> elemente
$(".test").hide(); // sakriva sve elemente sa klasom test
$("#test").hide(); // sakriva element sa id-om "test“
$("#test").text("tekst"); // menja test paragrafa
```

Selektori

- Analogno CSS-ovim selektorima
- Elementi zadatog imena:

```
$("p")
```

- Element zadatog id-a

```
$("#id")
```

- Elementi zadate klase:

```
$(".klasa")
$("p.klasa")
```

- Elementi sa atributima:

```
$("a[href]")
```

CSS selektori (podsetnik)

CSS selektor	Primer	Značenje
`*`	`$("*")`	svi elementi na stranici
`#id`	`$("#neki_id")`	element sa atributom id, čija je vrednost navedena iza znaka `#`
`.class`	`$(".neka_klasa")`	svi elementi koji pripadaju klasi čiji je naziv naveden iza znaka `.`
`element`	`$("p")`	svi elementi čiji *tag* ima navedeni naziv
`s1, s2, ...`	`$("p, th, .neka_klasa, #neki_id")`	višestruka selekcija; unija rezultata svih selektora, pojedinačni selektori se odvajaju znakom `,`
`parent child`	`$("li a")`	traženi potomci navedenog roditelja
`:first`	`$("ul li:first")`	prvi među elementima pogođenim selekcijom
`:last`	`$("ul li:last")`	poslednji među elementima pogođenim selekcijom

CSS selektori (podsetnik)

primer2.html

CSS selektor	Primer	Značenje
`:first-child`	`$("tr:first-child")`	element pogođen koji je prvi potomak svog roditelja
`:last-child`	`$("tr:last-child")`	element pogođen koji je poslednji potomak svog roditelja
`:not(s)`	`$("input:not(:checked)")`	svi elementi pogođeni selekcijom, na koje se ne odnosi selekcija u zagradi
`:odd`	`$("tr:odd")`	svi neparni elementi pogođeni selekcijom
`:even`	`$("tr:even")`	svi parni elementi pogođeni selekcijom
`:parent`	`$("#neki_id:parent")`	roditelj čvora pogođenog selekcijom
`[attribute]`	`$("input[checked]")`	svi elementi koji poseduju navedeni atribut
`[attribute=value]`	`$("input[type=text]")`	svi elementi koji poseduju navedenu vrednost atributa
i mnogi drugi…		

Selektori

- Da li je iz JavaScript-a

```
var contents = document.getElementById('contents');
```

- isto sa JQuery

```
var contents = $('#contents');
```

Selektori

- Da li je iz JavaScript-a

```
var contents = document.getElementById('contents');
```

- isto sa JQuery

```
var contents = $('#contents');
```

- Nije
- Prva će vratiti HTML DOM objekat
- Druga će vratiti jQuery objekat koji se vrapuje oko HTML DOM objekta i koji pruža jQuery metode.
- Nad pozivom $() mogu se pozvati JQuery metode kao css() ili animate()

Pristup HTML DOM objektu iz jQuery objekta

- Dobijanje HTML DOM objekata koji je vrapovan u JQuery objektu

```
$('#contents')[0]
$('#contents').get(0);
```

Rukovaoci događajima

- događaji se upotrebom *jQuery* biblioteke registruju direktno iz skripte u head sekciji
- time se HTML kod samog elementa rasterećuje od nepotrebnih *JavaScript* izraza
- tipično se *inline* kreira rukovaoc događajima – bezimena funkcija (tzv. *callback*), mada ona može biti definisana u bilo kom trenutku u skripti pre trenutka registrovanja, a registracija se onda može obaviti putem njenog naziva ili putem reference na funkciju
- Imena događaja su iz JavaScript domena (bez **on** prefiksa)

```
$(domObjekat).događaj(function() {
    …
})

$("CSS selektor")).događaj(function() {
    …
})
```

```
function funkcija1() {
    …
}

$(…).događaj(funkcija1)
```

```
var funkcija2 = function ()
{
    …
}

$(…).događaj(funkcija2)
```

Java Script Događaji

Događaj	Dešava se kada...
onabort	se prekine učitavanje slike
onblur	element izgubi fokus
onchange	korisnik pormeni sadržaj polja
onclick	se klikne mišem na objekat
ondblclick	se dva puta klikne po objektu
onerror	se dogodi greška prilikom učitavanja dokumenta ili slike
onfocus	element dobije fokus
onkeydown	se pritisne taster
onkeypress	se pritisne, pa otpusti taster, ili se drži pritisnut
onkeyup	se otpusti taster

Java Script Događaji

Događaj	Dešava se kada...
onload	se stranica ili slika učita
onmousedown	se pritisne dugme miša
onmousemove	se miš pomera
onmouseout	miš izađe izvan zone elementa
onmouseover	miš pređe preko elementa
onmouseup	se otpusti dugme miša
onreset	se klikne na reset dugme
onresize	se prozoru ili frejmu promeni veličina
onselect	je tekst selektovan
onsubmit	se klikne na dugme subit u formi
onunload	korisnik napusti stranicu
i mnogi drugi...	

jQuery Događaji

Događaj	Dešava se kada...
`$(…).ready(…)`	... se dokument učita; registruje se na *document* objekat
`$(…).click(…)`	... korisnik klikne na element
`$(…).submit(…)`	... korisnik *submit*-uje formu
`$(…).mouseover(…)`	... korisnik pređe mišem preko elementa
`$(…).focus(…)`	... korisnik uđe u polje za unos
`$(…).blur(…)`	... korisnik napusti polje za unos
`$(…).keydown(…)`	... korisnik pritisne taster
`$(…).keypress(…)`	... korisnik pritisne pa otpusti taster, ili drži taster
`$(…).keyup(…)`	... korisnik otpusti taster
i mnogi drugi...	

jQuery Događaji

primer3.html

Miš	Tastatura	Forma	Dokument
`click`	`keypress`	submit	load
`dbclick`	keydown	change	Resize
`mouseenter`	keyup	Focus	scroll
`mouseleave`		blur	unload
`hover`			

$(document).ready(…)

- *ready* događaj *document* objekta se odvija kada stranica završi učitavanje
- s obzirom na to da se rukovaoci događajima registruju direktno iz skripte, potrebno je sačekati da se svi elementi stranice učitaju pre registracije
- kod koji registruje ostale događaje zato mora da se nalazi unutar rukovaoca događajem *ready* objekta *document*
- ovaj pristup je koristan i za druge namene pri kojima se pristupa elementima stranice (npr. selekcija elemenata odmah po učitavanju stranice radi keširanja referenci u cilju ubrzanja performanse)

```
$(document).ready(function() {
    …
})
```

Metode

- metode pozivaju nad objektima i vraćaju objekte koji predstavljaju elemente
- *jQuery* preskače tesktualne čvorove i čvorove koji predstavljaju atribute i njima rukuje implicitno

Metode manipulacije elementom DOM stabla

- pristup i izmena elementa stranice

Naziv	Upotreba
`$(…).text()` `$(…).text(novaVrednost)`	čitanje tekstualnog sadržaja elementa izmena tekstualnog sadržaja elementa

`$(…).html()`
 	
`$(…).html(novaVrednost)`
	čitanje podstabla elementa u formi *string* vrednosti koja sadrži HTML kod podstabla
	izmena podstabla elementa upotrebom *string* literala ili promenljive koji sadrže HTML kod podstabla

`$(…).val()` `$(…).val(novaVrednost)`	čitanje vrednosti polja za unos izmena vrednosti polja za unos

`$(…).attr(nazivAtributa)`
`$(…).attr(nazivAtributa, novaVrednost)`
`$(…).removeAttr(nazivAtributa)`
	čitanje vrednosti atributa
	dodavanje atributa ili izmena vrednosti atributa
	uklanjanje atributa

- prilikom postavljanja vrednosti, može se proslediti i *callback* funkcija koja će da izračuna i vrati tu vrednost

Metode manipulacije elementom DOM stabla *primer4.html*

Naziv	Upotreba
`$(…).prop(nazivProperty-a)`	čitanje vrednosti dinamičkih atributa, npr. *checked*, *selected* i sl.

```
var checkboxAdministrator = $("input[name=administrator]")

…

// vratiće početnu vrednost atributa kakva je bila pri učitavanju stranice
if (checkboxAdministrator.attr("checked")) {
        …
}
// vratiće tekuću vrednost atributa (radiće očekivano)
if (checkboxAdministrator.prop("checked")) {
        …
}
// skraćeni zapis (radiće očekivano)
if (checkboxAdministrator.checked) {
        …
}
```

Kreiranje novog jQuery HTML elementa

- Kreiranje HTML elementa

```
// Create jQ element and its' inerHTML with HTML code
var txt1jQ = $("<p>Text.</p>");



// Create jQ element with HTML code and its' inerHTML with jQuery code
var  txt2jQ = $("<p></p>").text("Text.");



// Create DOM element with DOM manipulation JS code
var txt3Dom = document.createElement("p");
txt3.innerHTML = "Text.";
// Create jQ element from DOM
var  txt3jQ = $(txt3Dom);
```

Metode manipulacije DOM stablom

primer5.html

- umetanje i uklanjanje elemenata

Naziv	Upotreba

`$(…).append(…)`
`$(…).prepend(…)`
`$(…).empty()`
	dodaje element/elemente na kraj liste čvorova potomaka pozivajućeg čvora **(podnivo)**
	dodaje element/elemente na početak liste čvorova potomaka pozivajućeg čvora(**podnivo**)
	briše podstablo pozivajućeg elementa (**element ostaje**)

`$(…).after(…)`
`$(…).before(…)`
`$(…).remove()`
	dodaje element/elemente iza pozivajućeg čvora u istom nivou stabla (**isti nivo**)
	dodaje element/elemente ispred pozivajućeg čvora u istom nivou stabla (**isti nivo**)
	uklanja pozivajući element/elemente i sve njegove potomke (**ništa ne ostaje**)

- Metoda remove() može da ima argument, koji je tipa string i predstavlja selektor filtriranja
  - time se filtrira lista elemenata za uklanjanje (presek osnovnog selektora i argumenta funkcije

```
// svi p tagovi sa klasom 'italic’
$("p").remove(".italic");
```

Metode manipulacije DOM stablom

- Pronalaženje elementa na osnovu njegovog položaja u DOM stablu, terminologija:
  - Ancestor – čvor prethodnih
  - Descendant – čvor sledbenik
  - Parent – čvor direktni prethodnik
  - Child – čvor direktni sledbenik
  - Sibling – čvor na istom nivou

Metode manipulacije DOM stabolm

primer6.html

- vertikalno kretanje kroz stablo

Naziv	Upotreba

`$(…).parent()`
`$(…).parents()`
`$(…).parentsUntil(selektor)`
	vraća roditeljski element, direktni prethodnik, (**penjanje**)
	vraća sve roditeljske elemente, listu svih roditelja(Penjanje) , (**penjanje**)
	vraća sve roditeljske elemente do (a isključujući) roditelja selektovanog argumentom `selektor`, (**penjanje**)

`$(…).children()` `$(…).find(selektor)`	vraća sve elemente potomke, (**silazak**) pronalazi element/elemente u podstablu pozivajućeg elementa koji su identifikovani argumentom kao selektorom, svi sledbenici selektovani zadatim kriterijumom `selektor`, (**silazak**)

Metode manipulacije DOM stablom

primer7.html

- horizontalno kretanje kroz stablo

Naziv	Upotreba
`$(…).siblings()`	vraća sve elemente na istom nivou (koji imaju zajedničkog roditelja sa pozivajućim elementom)

`$(…).next()`
`$(…).nextAll()`
`$(…).nextUntil(selektor)`
	vraća element na istom nivou neposredno iza pozivajućeg elementa
	vraća sve elemente na istom nivou iza pozivajućeg elementa
	vraća sve elemente na istom nivou iza pozivajućeg elementa do (a isključujući) elementa pogođenog selekcijom

`$(…).prev()`
`$(…).prevAll()`
`$(…).prevUntil(selektor)`
	vraća element na istom nivou neposredno pre pozivajućeg elementa
	vraća sve elemente na istom nivou pre pozivajućeg elementa
	vraća sve elemente na istom nivou pre pozivajućeg elementa do (a isključujući) elementa pogođenog selekcijom

Metode manipulacije elementom DOM stabla *primer8.html*

- rukovanje CSS klasama i CSS svojstvima

Naziv	Upotreba

`$(…).addClass(…)`
`$(…).removeClass(…)`
`$(…).toggleClass(…)`
	dodaje klasu pozivajućem elementu
	uklanja klasu pozivajućeg elementa
	dodaje klasu pozivajućeg elementa ukoliko je već ne poseduje i obrnuto

`$(…).css(svojstvo, vrednost)`	postavlja CSS svojstvo elementa na novu vrednost

Metode manipulacije elementom DOM stabla *primer9.html*

- Efekti
  hide(), show() i toggle()
- animacija sklanjanje/pojavljivanje selektovanog elementa tako što se rapidno vrši promena propertija CSS **opacity** i CSS **height** od elementa

Naziv	Upotreba
`$(…).hide(…)`	sakriva element/elemente
`$(…).show(…)`	prikazuje element/elemente
`$(…).toggle(…)`	sakriva element/elemente ako je/su bio/bili prikazan/prikazani i obrnuto

- Moguće je zadati brzinu, kao i callback
  - `.hide();`
  - `.hide(1000);`
  - `.hide('slow'); //'slow', 'normal', or 'fast'`
  - `.hide(speed,callback);`

Metode manipulacije elementom DOM stabla

- Efekti
  fadeIn(), fadeOut(), fadeToggle(), fadeTo()
- animacija sklanjanje/pojavljivanje selektovanog elementa tako što se rapidno vrši promena propertija CSS **opacity** od elementa

Naziv	Upotreba
`$(…).fadeIn(…)`	animirano prikazuje element/elemente
`$(…).fadeOut(…)`	animirano sakriva element/elemente
`$(…).fadeToggle(…)`	animirano sakriva element/elemente ako je/su bio/bili prikazan/prikazani i obrnuto
`$(…).fadeTo(…)`	animirano sakriva ili prikazuje element/elemente do zadate providnost (od 0 do 1)

- Moguće je zadati brzinu, kao i callback
  - `.fadeIn();`
  - `.fadeIn(1000);`
  - `.fadeIn('slow'); //'slow', 'normal', or 'fast'`
  - `.fadeIn(`*`speed,callback`*`);`
  - `.fadeTo(`*`speed,opacity`*`);`
  - `.fadeTo(`*`speed,opacity,callback`*`);`

Metode manipulacije elementom DOM stabla *primer10.html*

- Efekti
  slideUp(…), slideDown(…), slideToggle()
- animacija sklanjanje/pojavljivanje selektovanog elementa tako što se rapidno vrši promena propertija CSS **height** od elementa

Naziv	Upotreba
`$(…).slideUp(…)`	animirano sakriva element/elemente
`$(…).slideDown(…)`	animirano prikazuje element/elemente
`$(…).slideToggle(…)`	animirano sakriva element/elemente ako je/su bio/bili prikazan/prikazani i obrnuto

- Moguće je zadati brzinu, kao i callback
  - `.slideUp();`
  - `.slideUp(1000);`
  - `.slideUp('slow'); //'slow', 'normal', or 'fast'`
  - `.slideUp(`*`speed,callback`*`);`

Metode manipulacije elementom DOM stabla

- Efekti
  animate(…)
- animacija elementa tako što se rapidno vrši promena propertija CSS od elementa

Naziv	Upotreba
`$(…).animate(…)`	proizvoljna animacija u odnosu na zadate parametre

- Moguće je zadati brzinu, kao i callback
  - `.animate({params});`
  - `.animate({params}, speed);`
  - `.animate({params}, speed, callback);`

```
$("button").click(function(){
  $("div").animate({left:'250px'});
});
```

```
$("button").click(function(){
  $("div").animate({
    left:'250px',
    height:'+=150px',
    width:'+=150px'
  });
});
```

Metode

primer11.html

- Filteri – vraća elemet/elemente iz liste selektovanih elemenata

Naziv	Upotreba
`$(selektor).first()` `$(selektor).last()`	vraća prvi među elementima pogođenim prethodnom selekcijom vraća poslednji među elementima pogođenim prethodnom selekcijom

`$(selektor).filter(užiSelektor)`
 	
`$(selektor).not(užiSelektor)`
	vraća elemente pogođene dodatnom selekcijom među elementima pogođenim prethodnom selekcijom
	vraća elemente koji ne zadovoljavaju dodatnu selekciju među elementima pogođenim prethodnom selekcijom

`$(selektor).eq(index)`	vraća element na zadatoj poziciji među elementima pogođenim prethodnom selekcijom

Razlika između jQ metoda filter i find

Metode

primer12.html

Naziv	Upotreba
`$(…).each(...)`	iteracija kroz više čvorova obuhvaćenih css selektorom

- Funkcija each() prolazi kroz listu elemenata i za svaki izvršava opcioni kod prosleđen kao funkcija

```
var checkboxovi = $("input[type=checkbox]")

…
// neće raditi jer je checkboxovi jedan jQuery objekat!
for (var it in checkboxovi) {
        …
}
// radiće očekivano; it je indeks petlje, a checkbox poprima jedan po jedan DOM čvor
checkboxovi.each(function(it, checkbox) {
        // svaki čvor se mora ponovo wrap-ovati u jQuery objekat
        if ($(checkbox).checked) {
                …
        }
})
```

Callbacks

nastavak naredni cas

primer13.html

- ako jedan *jQuery* poziv traje određeno vreme (npr. animacija), moguće je odložiti sledeći *jQuery* poziv dok to vreme ne istekne, ako se on smesti u *callback* i doda kao poslednji argument prvog poziva
- Callback je funkcija koja se poziva nakon obavljenog posla (sekvencionalno)
- Obično je to parametar neke funkcije
- U okviru collback fukcije može se koristiti i $(this) da bi se pristupilo elementu za koji se izvršava callback funkcija

```
$("button").click(function() {
    $("p").slideToggle() // 1. poziv

    var dugme = $(this)
    if (dugme.text() == "Sakrij") {
        dugme.text("Otkrij") // 2. poziv
    } else {
        dugme.text("Sakrij") // 2. poziv
    }
})
```

1 i 2 pozivi izvršvaće se paralelno

callback

```
$("button").click(function() {
    var dugme = $(this)
    $("p").slideToggle(function() {  // 1. poziv
        if (dugme.text() == "Sakrij") {
            dugme.text("Otkrij") // 2. poziv unutar callback-a
        } else {
            dugme.text("Sakrij") // 2. poziv unutar callback-a
        }
})
```

Prvo će se izvršiti 1 a posle 2 poziv (sekvencijalno)

Ulančavanje

primer14.html

- *jQuery* pozivi se mogu ulančavati u istoj naredbi (kad je to primenljivo)

$(selector).efekat().efekat();

- Izvršavanje više jQuery komandi jedne za drugom nad istim elementom se izvršava paralelno, osim ako komande nisu suprotne tada se izvršava sekvencionalno

```
// pristupalo bi se elementima pre nego što su učitani
$(document).ready(function () {// sačekati da se stranica učita
    $("button#b_1").click(function () {
        // sekvencijalno - oboji crveno i sakri, pa zatim pokazi
        $("p").css("color", "red").hide(1000).show(1000);
    });
    $("button#b_2").click(function () {
        // paralelno - oboji zeleno i sakri
        $("p").css("color", "blue").hide(1000).css("color", "green");
    });
});
```

Ulančavanje

primer15.html

- Dozvoljeno je korišćenje callback funkcija pri ulančavanju
- Voditi računa da će callback funkcija da se izvrši tek po zvršetku efekta u kome je callback prosleđen

```
$("button").click(function() {
    var dugme = $(this)
    var pasus = $("p")
    pasus.css("color", "red").slideToggle(function() { // ulančavanje
        pasus.css("color", "black")
        if (dugme.text() == "Sakrij") {
            dugme.text("Otkrij")
        } else {
            dugme.text("Sakrij")
        }
    })
})
```

Više efekata

- Efekti se mogu izvršavati paralelno ili sekvencijalno
- Zavisi od toga koji se efekti ulančavaju nad selektovanim elementom/elementima, da li su efekti suprotni ili ne, i da li se pri pozivu efekta koristi i callback funkcija

Dinamičko dodavanje elemenata

primer16.html

- Pokazni primer kako se može dinamički dodati ili ukoniti red iz tabele u formi za plaćanje parkinga

Dinamičko dodavanje elemenata

- Primena jQuery za potrebe lokalizacije u Bioskop web aplikaciji

BioskopVebAplikacijaT9, zanrovi.html, base.html, lokacijaJS.js

Dodatni materijali

- https://api.jquery.com/
- https://www.tutorialspoint.com/jquery/jquery-overview.html
- https://www.tutorialrepublic.com/jquery-tutorial/
- https://www.w3schools.com/jquery/

- https://www.w3schools.com/js/
- https://www.w3schools.com/js/js_object_definition.asp