jQuery selektori i događaji

U prethodnoj lekciji izneti su osnovni postulati jQuery biblioteke. Prikazan je i prvi primer korišćenja jQuery biblioteke za realizaciju realnog primera manipulacije DOM strukturom. Za realizaciju primera je upotrebljeno nekoliko elemenata jQuery biblioteke, čije je razumevanje neophodno kako bi se razumeo kompletan primer. U prethodnoj lekciji smo započeli upoznavanje takvih osnovnih pojmova jQuery biblioteke. U ovoj lekciji nastavljamo dalje, pa će stoga redovi koji slede biti posvećeni različitim pristupima za selektovanje elemenata i obradu događaja korišćenjem jQueryja.

Osnovni jQuery selektori

U prethodnoj lekciji započeli smo priču o osnovnim jQuery selektorima. U ovoj lekciji će prvo biti obnovljeno ono što već znamo, a zatim će biti prikazani i neki napredni scenariji selektovanja.

Selektovanje elemenata DOM strukture korišćenjem jQueryja obavlja se upotrebom jedne od specijalnih funkcija:

- `$()`
- `jQuery()`

Dve prikazane funkcije imaju identičan efekat i na vama je da odabere koju ćete koristiti. Moguće je čak u jednom istom dokumentu koristiti obe prikazane funkcije. Ipak, u nekim situacijama se može javiti potreba za korišćenjem baš funkcije `jQuery()`. Naime, jQuery nije jedina biblioteka koja poseduje funkciju sa nazivom `$`. Ukoliko se unutar jednog dokumenta koristi još neka biblioteka koja poseduje ovu funkciju, može doći do konflikata. U takvim situacijama mora se koristiti funkcija `jQuery()`. Ovo je inače i osnovni razlog postojanja alternativne `jQuery()` funkcije. Ukoliko znate da će se unutar dokumenta koristiti samo jQuery biblioteka, slobodno možete koristiti funkciju `$()`.

Funkcije za selektovanje DOM elemenata prihvataju jedan parametar kojim se definiše kriterijum na osnovu koga će biti obavljeno selektovanje jednog ili više elemenata. Prilikom definisanja parametra koji se prosleđuje jQuery funkcijama za selektovanje, primenjuju se pravila CSS jezika za definisanje selektora. Tako je moguće definisati sledeća tri tipa osnovnih jQuery selektora:

- #id selektor
- tipski selektor
- .class selektor

#id selektor:

```
$("#heading-1")
```

Korišćenjem selektora na osnovu vrednosti `id` atributa, uvek se obavlja selektovanje samo jednog elementa. Tako se prikazani selektor može koristiti za selektovanje ovakvog HTML elementa:

```
<h1 id="heading-1"> This is heading 1 </h1>
```

Tipski selektor:

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

Tipskim selektorom obavlja se selektovanje svih elemenata jednog tipa. U prikazanom primeru je za tip naveden HTML element p. Tako se prikazani pristup može koristiti za selektovanje svih elemenata iz sledeće strukture:

```
<p>Paragraph 1</p>
<p>Paragraph 2</p>
<p>Paragraph 3</p>
```

.class selektor:

```
$(".my-class")
```

Klasnim selektorom se selektuju svi elementi koji poseduju definisanu klasu. U prikazanom primeru je navedena klasa `.my-class`.

```
<p class="my-class my-other-class">Paragraph 1</p>
<p class="my-class">Paragraph 2</p>
<p>Paragraph 3</p>
```

Na primeru ovakve HTML strukture, prikazani klasni jQuery selektor obaviće selektovanje prvog i drugog paragrafa, s obzirom na to da oni poseduju klasu `.my-class`.

Pitanje

Odabrati jQuery funkcije za selektovanje:

a) $()
b) javaScript()
c) &()

Objašnjenje:

Selektovanje elemenata DOM strukture korišćenjem jQueryja obavlja se upotrebom jedne od specijalnih funkcija $().

Napredni i složeni jQuery selektori

Pored tri upravo prikazana osnovna selektora, prilikom selektovanja elemenata korišćenjem jQueryja moguće je koristiti i sve ostale selektore koji postoje u CSS jeziku. Takođe, različite selektore je moguće kombinovati, čime se dobijaju složeni selektori. U nastavku ovog poglavlja biće prikazano nekoliko primera takvih naprednih i složenih selektora.

Univerzalni selektor

Na primer, za selektovanje svih elemenata jednog dokumenta moguće je napisati:

```
$("*")
```

Kao parametar metode za selektovanje sada je postavljen karakter zvezdica (*). Iz CSS-a je poznato da se ovakav selektor naziva univerzalni, što praktično znači da se njime selektuju svi HTML elementi unutar dokumenta.

Selektovanje korišćenjem pseudoklasa

jQuery omogućava i korišćenje pseudoklasa prilikom selektovanja, pa je tako moguće napisati nešto ovako:

```
$("p:first")
```

Na ovaj način biće selektovan prvi paragraf element unutar dokumenta.

Selektovanje na osnovu prisustva atributa

Moguće je selektovati i elemente na osnovu prisustva određenog atributa:

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

Na ovaj način biće selektovani svi elementi koji na sebi poseduju `href` atribut.

Selektovanje na osnovu vrednosti atributa

Ukoliko je, pored prisustva nekog atributa na elementu, potrebno izvršiti selektovanje i na osnovu vrednosti, dovoljno je napisati:

```
$("a[target='_blank']")
```

Na ovaj način se selektuju svi `a` elementi koji za vrednost atributa `target` imaju `_blank`.

Kombinovanje selektora

Prilikom formulisanja jQuery selektora moguće je kombinovati više pojedinačnih selektora, baš kao u CSS jeziku. Tako je moguće napisati:

```
$("#my-div *")
```

Na ovaj način biće selektovani svi HTML elementi koji se nalaze unutar elementa sa `id`-jem `my-div`.

Ukoliko je potrebno selektovati neki element na osnovu više kriterijuma, moguće je napisati:

```
$("p.intro")
```

Na ovaj način se selektuju svi paragraf elementi koji na sebi imaju klasu `.intro`.

Metode za pretragu DOM stabla

Nakon selektovanja jednog ili više elemenata korišćenjem jQueryja, moguće je iskoristiti neku od ugrađenih metoda koje omogućavaju da se DOM stablo pretraži u odnosu na selektovani element. Tako jQuery omogućava sledeće:

- pretragu roditeljskih elemenata;
- pretragu elemenata naslednika (dece);
- pretragu srodnih elemenata.

Za obavljanje navedenih pretraga koristi se nekoliko različitih metoda koje će biti ilustrovane u nastavku. Sve one će biti prikazane na primeru sledeće HTML strukture:

```
<body>
    <section id="section1">
        <h1 id="heading1"> This is heading 1</h1>
        <p class="my-class">Paragraph 1</p>
        <p class="my-class">Paragraph 2</p>
        <p>Paragraph 3</p>
        <ul id="my-list">
            <li>one</li>
            <li>two</li>
            <li>three</li>
        </ul>
    </section>
</body>
```

Svi primeri u nastavku biće definisani unutar upravo prikazanog HTML dokumenta.

Pretraga roditelja

Za dobijanje roditelja nekog elementa moguće je koristiti metode ilustrovane tabelom 11.1.

Metoda	Opis
`parent()`	vraća direktan roditeljski element
`parents()`	vraća roditeljske elemente sa svih nivoa, sve do korenog elementa HTML dokumenta (`html`)
`parentsUntil()`	vraća sve roditeljske elemente između elementa nad kojim se poziva i elementa koji je ovoj metodi prosleđen kao parametar

Tabela 11.1. Metode za dobijanje roditeljskih elemenata

Primer korišćenja **parent()** metode može da izgleda ovako:

```
let elem = $("li:first-child").parent();
console.log(elem.attr('id'));
```

U primeru je prvo obavljeno selektovanje svih `li` elemenata koji su prvi potomci svojih roditelja. U referentom HTML dokumentu postoji samo jedan takav element, odnosno prvi `li` element unutar neuređene liste sa `id`-jem `my-list`. Nad tako selektovanim `li` elementom poziva se `parent()` metoda, čime se dobija direktan roditelj takvog elementa. Drugom naredbom se ispisuje `id` tako dobijenog roditeljskog elementa i unutar konzole se dobija:

```
my-list
```

Metoda attr()

U primeru je prvi put upotrebljena jQuery funkcija `attr()`. Reč je o funkciji koja omogućava čitanje vrednosti nekog svojstva koje se nalazi na elementu.

Za razliku od funkcije `parent()`, funkcija **parents()** vraća sve roditelje nekog elementa:

```
let elements = $("li:first-child").parents();

elements.each(function(){
   console.log($(this).prop('nodeName'));
});
```

Sada je nad objektnom reprezentacijom prvog `li` elementa pozvana `parents()` jQuery metoda. S obzirom na to da se ovom metodom dobijaju roditelji sa svih nivoa, u nastavku je iskorišćen još jedan specifičan pristup jQuery biblioteke za prolazak kroz niz elemenata.

Metoda each()

Metoda `each()` koristi se za prolazak kroz niz selektovanih DOM objekata unutar jQuery objekta. Tako se u prikazanom primeru ova metoda koristi za prolazak kroz sve DOM objekte koji predstavljaju roditeljske elemente prethodno selektovanog `li` elementa.

Unutar metode `each()`, za pristup pojedinačnim DOM elementima je moguće koristiti ključnu reč `this`. Obratite pažnju na to da se ključna reč `this` odnosi na izvorni DOM objekat, pa se zbog toga ona u primeru postavlja kao parametar jQuery funkcije za selektovanje. Na taj način se svaki izvorni DOM objekat ponovo prevodi u jQuery objekat, što na kraju omogućava da se nad takvim objektima pozovu specifične jQuery metode.

U prikazanom primeru se obavlja ispis tipova svih selektovanih roditeljskih elemenata. To se postiže još jednom specifičnom jQuery metodom – `prop()`.

Metoda prop()

Metoda `prop()` omogućava da se na veoma lak način obavi čitanje nekog od svojstava izvornog DOM objekta. Ovoj metodi se prosleđuje naziv svojstva čiju vrednost je potrebno pročitati. U primeru se metodi `prop()` prosleđuje vrednost `nodeName`, čime je rečeno da želimo da dobijemo vrednosti `nodeName` svojstva.

Pošto je `nodeName` svojstvo izvornih DOM objekata, a ključna reč `this` unutar `each()` metode odnosi se upravo na takve objekte, nazivi tagova su mogli biti dobijeni i korišćenjem čistog JavaScript koda:

```
elements.each(function(){
   console.log(this.nodeName);
});
```

Unutar konzole se dobijaju tipovi svih roditeljskih elemenata:

```
UL
SECTION
BODY
HTML
```

Na kraju, selektovanje roditeljskih elemenata je moguće obaviti i korišćenjem metode **parentsUntil()**. Ova metoda nam omogućava da ograničimo selektovanje roditelja, definisanjem elementa do kog će se selektovanje obavljati:

```
let elements = $("li:first-child").parentsUntil("body");

elements.each(function(){
   console.log(this.nodeName);
});
```

Sada je metoda `parents()` zamenjena metodom `parentsUntil()`, kojoj je prosleđena vrednost `body`. Unutar konzole se sada dobija sledeći ispis:

```
UL
SECTION
```

Na osnovnu dobijenog ispisa, može se zaključiti da metoda `parentsUntil()` selektuje sve elemente između elementa nad kojim se poziva i elementa koji se prosleđuje kao parametar.

Pretraga naslednika (dece)

Prethodni redovi su ilustrovali metode za pretragu roditelja. jQuery obezbeđuje skup metoda koje je moguće koristiti za pretragu elemenata naslednika, odnosno dece (tabela 11.2).

Metoda	Opis
`children()`	vraća direktne potomke nekog elementa, odnosno potomke samo sa jednog nivoa ispod
`find()`	vraća potomke sa svih nivoa

Tabela 11.2. Metode za dobijanje elemenata naslednika

Primer korišćenja **children()** metode može da izgleda ovako:

```
let elements = $("#section1").children();

elements.each(function () {
   console.log(this.nodeName);
});
```

U primeru je nad elementom sa `id`-jem `section1` pozvana metoda `children()`. Unutar konzole se dobija sledeći ispis:

```
H1
P
P
P
UL
```

Reč je o tipovima svih elemenata koji su direktni potomci `#section1` elementa. Metodi `children()` se može proslediti i jedan parametar i tada se selektuju samo direktni potomci koji zadovoljavaju definisan kriterijum:

```
let elements = $("#section1").children("p");

elements.each(function () {
   console.log(this.nodeName);
});
```

Sada je kao parametar metode `children()` navedeno `p`, što će kao rezultat stvoriti selektovanje samo paragrafa koji su direktni potomci elementa `section1`:

```
P
P
P
```

Metodom **find()** selektuju se svi potomci nekog elementa, bilo da su oni direktni ili indirektni:

```
let elements = $("#section1").find("*");

elements.each(function () {
   console.log(this.nodeName);
});
```

Za razliku od metode `children()`, metodi `find()` je neophodno proslediti neki parametar. U primeru joj je prosleđen univerzalni selektor (zvezdica), pa se na ovaj način unutar konzole dobija:

```
H1
P
P
P
UL
LI
LI
LI
```

Parametrom koji se prosleđuje metodi `find()` obavlja se filtriranje potomaka koji će biti selektovani, odnosno, definiše se uslov selekcije:

```
let elements = $("#section1").find("li");

elements.each(function () {
   console.log(this.nodeName);
});
```

Sada je metodi `find()` prosleđen parametar `li`, što će za rezultat imati selektovanje samo potomaka koji su tipa `li`:

```
LI
LI
LI
```

Pretraga srodnika

Pored potomaka i naslednika, jQuery omogućava i selektovanje srodnih elemenata. Reč je o elementima koji se unutar DOM stabla nalaze na istom nivou, unutar jednog roditelja. Metode koje se mogu koristiti za pretragu srodnika prikazane su tabelom 11.3.

Metoda	Opis
`siblings()`	vraća sve srodne elemente nekog elementa
`next()`	vraća naredni srodni element
`nextAll()`	vraća sve naredne srodne elemente
`nextUntil()`	vraća naredne srodne elemente, sve do elementa definisanog parametrom
`prev()`	vraća prvi prethodni element
`prevAll()`	vraća sve prethodne elemente
`prevUntil()`	vraća prethodne elemente, sve do elementa definisanog parametrom

Tabela 11.3. Metode za pretragu srodnih elemenata

Svi srodni elementi nekog elementa, bez obzira na to da li se u strukturi elemenata nalaze pre ili posle njega, mogu se dobiti metodom **siblings()**:

```
let elements = $("#my-list").siblings();

elements.each(function () {
   console.log($(this).prop('nodeName'));
});
```

Nad objektom koji je dobijen selektovanjem elementa sa `id`-jem `my-list` pozvana je metoda `siblings()`. Na ovaj način unutar konzole se dobija:

```
H1
P
P
p
```

Potpuno je jasno da su na ovaj način selektovani svi srodni elementi neuređene liste, odnosno svi elementi koji se pored takve liste nalaze unutar `#section1` elementa.

Ukoliko je potrebno selektovati samo prvi naredni srodni element, koristi se metoda **next()**:

```
let elements = $("#heading1").next();

elements.each(function () {
   console.log($(this).prop('nodeName'));
});
```

Sada je nad objektom elementa koji je predstavljen `id`-jem `heading1` pozvana funkcija `next()`. Unutar konzole se dobija tip prvog narednog srodnog elementa:

```
P
```

Svi naredni srodni elementi se mogu dobiti metodom **nextAll()**:

```
let elements = $("#heading1").nextAll();

elements.each(function () {
   console.log($(this).prop('nodeName'));
});
```

Unutar konzole se dobija:

```
P
P
P
UL
```

Ukoliko je potrebno postaviti granični element do koga će se selektovanje srodnih narednih elemenata obavljati, koristi se metoda **nextUntil()**:

```
let elements = $("#heading1").nextUntil("ul");

elements.each(function () {
   console.log($(this).prop('nodeName'));
});
```

Sada se unutar konzole dobijaju tipovi svih narednih srodnih elemenata, do elementa tipa `ul`:

```
P
P
P
```

Upravo prikazane metode `next()`, `nextAll()` i `nextUntil()` selektuju naredne srodne elemente. Ukoliko je potrebno obaviti selektovanje prethodnih srodnih elemenata, koristi se skup metoda sa prefiksom prev – `prev()`, `prevAll()` i `prevUntil()`.

Prvi prethodni srodni element može se dobiti korišćenjem metode `prev()`:

```
let elem = $("#my-list").prev();
console.log(elem.text());
```

Metoda `prev()` pozvana je nad objektnom reprezentacijom `#my-list` elementa. Zatim je obavljen ispis teksta tako dobijenog elementa:

```
Paragraph 3
```

Metoda text()

Metoda `text()` je jQuery metoda koju je moguće koristiti za dobijanje teksta elementa.

Ukoliko je potrebno dobiti sve prethodne srodne elemente, koristi se metoda **prevAll()**:

```
let elements = $("#my-list").prevAll();

elements.each(function () {
   console.log($(this).prop('nodeName'));
});
```

Sada se unutar konzole dobijaju sledeći tipovi:

```
P
P
P
H1
```

Na kraju, opseg selektovanja prethodnih srodnih elemenata je moguće ograničiti, korišćenjem metode **prevUntil()**:

```
let elements = $("#my-list").prevUntil('h1');

elements.each(function () {
   console.log($(this).prop('nodeName'));
});
```

Unutar konzole se dobija:

```
P
P
P
```

Filtriranje elemenata

Korišćenjem jQuery funkcija za selektovanje, dobija se jedan jQuery objekat unutar koga može biti jedan ili više konkretnih DOM objekata. Drugim rečima, jQuery selektovanje može da podrazumeva dobijanje reference na jedan ili više elemenata. Kada selektovanje podrazumeva veći broj elemenata, moguće je koristiti nekoliko jQuery metoda za filtriranje dobijanih elemenata (tabela 11.4).

Metoda	Opis
`first()`	vraća prvi element iz skupa elemenata
`last()`	vraća poslednji element iz skupa elemenata
`eq()`	vraća element sa određene pozicije unutar skupa selektovanih elemenata
`filter()`	vraća elemente koji zadovoljavaju definisan kriterijum koji se prosleđuje kao parametar
`not()`	vraća sve elemente koji ne zadovoljavaju kriterijum koji se prosleđuje kao parametar

Tabela 11.4. Metode za filtriranje skupa selektovanih elemenata

Za demonstraciju mogućnosti metoda za filtriranje iz tabele 11.4, prvo je neophodno obaviti selekciju većeg broja DOM elemenata, korišćenjem pristupa koji su već ilustrovani:

```
let elements = $("#section1").children();
```

Na ovaj način je prvo obavljeno selektovanje elementa sa `id`-jem `section1`. Zatim se korišćenjem metode `children()` došlo do svih njegovih naslednika. Tako je unutar promenljive `elements` smešten jQuery objekat unutar koga se nalazi nekoliko izvornih DOM objekata.

Ukoliko je potrebno dobiti samo prvi element iz skupa selektovanih elemenata, dovoljno je iskoristiti metodu **first()**:

```
let firstElem = elements.first();
console.log(firstElem.text());
```

Na ovaj način se unutar promenljive `firstElem` smešta referenca na prvi element unutar selektovane DOM strukture. Stoga se unutar konzole dobija:

```
This is heading 1
```

Kada je potrebno dobiti samo poslednji od selektovanih elemenata, koristi se metoda **last()**:

```
let lastElem = elements.last();
console.log(lastElem.text());
```

Unutar konzole će biti dobijene vrednosti stavki neuređene liste, zato što se na ovaj način selektuje poslednji element, odnosno element `ul`:

```
one
two
three
```

jQuery omogućava da se dobije i element na tačno određenoj poziciji unutar selektovane strukture. Za obavljanje takvog posla se koristi metoda **eq()**, kojoj se prosleđuje indeks elementa:

```
let specificElem = elements.eq(1);
console.log(specificElem.text());
```

Bitno je obratiti pažnju na to da indeksiranje započinje od nule (0), pa se zbog toga prikazanim kodom unutar konzole dobija tekst prvog paragrafa:

```
Paragraph 1
```

Skup selektovanih elemenata se može filtrirati na osnovu sopstvenih kriterijuma, korišćenjem metode **filter()**:

```
let onlyParagraphs = elements.filter("p");

onlyParagraphs.each(function () {
   console.log($(this).text());
});
```

Metoda `filter()` je u primeru iskorišćena za filtriranje selektovanih elemenata. Njoj je prosleđen parametar `p`, čime je rečeno da želimo da dobijemo samo paragraf elemente iz skupa selektovanih elemenata. Zbog toga se unutar konzole dobija:

```
Paragraph 1
Paragraph 2
Paragraph 3
```

Ukoliko je elemente potrebno filtrirati na osnovnu nezadovoljavanja nekog kriterijuma, moguće je koristiti metodu **not()**:

```
let everythingExceptParagraphs = elements.not("p");
everythingExceptParagraphs.each(function () {
   console.log($(this).text());
});
```

Sve je isto kao u prethodnom primeru, osim funkcije koja se poziva nad skupom elemenata. Nad skupom elemenata sada se poziva metoda `not()`, čime će filtriranje biti obavljeno tako da će novi skup elemenata sadržati sve elemente osim paragrafa. Stoga se unutar konzole dobija tekst naslova i neuređene liste:

```
This is heading 1
one
two
three
```

Obrada događaja korišćenjem biblioteke jQuery

Obrada događaja je još jedan aspekt programiranja koji je znatno olakšan korišćenjem jQuery biblioteke. jQuery obezbeđuje tri različita mehanizma za pretplatu na događaje:

- korišćenje metoda sa imenima događaja
- korišćenje specijalne metode `on()`
- korišćenje specijalne metode `one()`

Korišćenje metoda sa imenima događaja

jQuery poseduje veliki broj metoda čija imena odgovaraju događajima za koje se želi izvršiti pretplata. Takvim metodama se prosleđuje funkcija koja se aktivira kada dođe do pojave definisanog događaja.

Na primer, kako bi se definisala logika koja će se aktivirati prilikom klika na jedan `button` element, dovoljno je uraditi sledeće:

```
<button id="my-button">Click me</button>
<script src="js/jquery-3.4.1.min.js"></script>
<script>
   $("#my-button").click(function () {
      alert("The element was clicked.");
   });
</script>
```

Prikazani primer predstavlja isečak jednog HTML dokumenta. Unutar `body` dela nalazi se jedan `button` element koji na sebi ima `id` atribut sa vrednošću `my-button`. Unutar `script` taga napisan je kod za definisanje logike koja će se aktivirati kada korisnik klikne na ovakav `button` element. Prvo je obavljeno selektovanje `button` elementa, korišćenjem jQuery funkcije. Nad selektovanim elementom zatim je pozvana metoda `click()`, kojom se definiše funkcija koja će se izvršiti kada dođe do klika na element. Takva logika je definisana kao anonimna funkcija, unutar koje je postavljen kod kojim se prikazuje jedan modalni prozor sa prigodnom porukom.

Prikazani jQuery pristup za pretplatu na `click` događaj ekvivalentan je sledećem čistom JavaScript kodu:

```
let myButton = document.getElementById("my-button");
myButton.addEventListener("click", function () {
   alert("The element was clicked.");
});
```

Pored upravo prikazane `click()` metode, veoma korisna je i jQuery metoda `ready()`. Ona omogućava definisanje logike koja će se izvršiti onda kada web pregledač izgradi kompletan DOM. To je veoma bitna činjenica, o čijoj važnosti je već bilo govora u jednoj od prethodnih lekcija. Naime, pretplata na događaje različitih DOM elemenata uopšte i nije moguća dok web pregledač ne izgradi objektnu strukturu dokumenta. Kako bismo bili sigurni da je takva struktura izgrađena, moguće je koristiti metodu `ready()`:

```
$(document).ready(function () {
   //code to be executed when DOM is ready
});
```

Bitno je primetiti da je metoda `ready()` pozvana nad `document` promenljivom koja je upakovana u jedan jQuery objekat, smeštanjem unutar selektorske funkcije. Metoda `ready()` prihvata jedan parametar, kojim se definiše funkcija sa logikom koja će se izvršiti kada web pregledač izgradi DOM. Sada je unutar ovakve anonimne funkcije moguće pisati kod za pristup elementima DOM strukture, bez bojazni da li je takva struktura izgrađena:

```
$(document).ready(function () {
   $("#my-button").click(function () {
      alert("The element was clicked.");
   });
});
```

Nešto ranije prikazani kod za pretplatu na `click` događaj `button` elementa smešten je unutar funkcije koja se aktivira kada pregledač izgradi DOM. Sada je ovakav kod moguće smestiti bilo gde unutar `body` dela dokumenta, pa čak i pre `button` HTML elementa koji ovakav kod koristi.

Metoda on() za pretplatu na događaje

Registrovanje logike koja će se aktivirati prilikom pojave događaja korišćenjem jQuery biblioteke se veoma lako može postići i korišćenjem metode `on()`. Tako se primer obrade `click` događaja iz prethodnih redova sada može transformisati na sledeći način:

```
$("#my-button").on("click", function () {
   alert("The element was clicked.");
});
```

Metodi `on()` prosleđena su dva parametra:

- **događaj** – definiše jedan ili više događaja
- **funkcija za obradu događaja** – definiše logiku koja će se izvršiti kada dođe do pojave događaja

Metoda `on()` ima dve prednosti u odnosu na korišćenje nešto ranije prikazanih metoda:

- korišćenjem metode `on()` moguće je u istom trenutku pretplatiti se na veći broj događaja
- pretplate obavljene korišćenjem metode `on()` veoma lako je moguće otkazati korišćenjem jQuery metode `off()`

Pretplata na veći broj događaja odjednom se može obaviti na sledeći način:

```
$("#my-button").on({
   mouseenter: function () {
      console.log("mouse entered button");
   },
   mouseleave: function () {
```

```
        console.log("mouse left button");
    },
    click: function () {
        console.log("button clicked");
    }
});
```

Sada je metodi `on()` prosleđen jedan anonimni objekat. On je kreiran korišćenjem vitičastih zagrada unutar kojih su definisani parovi svojstava i vrednosti. Svojstva su nazivi događaja, a vrednosti su anonimne funkcije kojima se definiše šta će se dogoditi prilikom pojave određenog događaja. Tako je prikazanim kodom obavljena pretplata na tri različita događaja `#my-button` elementa.

Još jedna od prednosti `on()` metode jeste mogućnost otkazivanja pretplate koja je na taj način definisana:

```
$("#my-button").on("click", function () {
    alert("The element was clicked.");
    $(this).off();
});
```

U primeru je prvo obavljena pretplata na `click` događaj `#my-button` elementa. Unutar funkcije koja se aktivira prilikom klika, definisana je poruka koja će se pojaviti u modalnom prozoru, ali i naredba za otkazivanje pretplate za sve događaje na `#my-button` elementu. Na ovaj način, modalni prozor sa porukom biće prikazan samo prilikom prvog klika na `button` element. Prilikom prvog klika aktiviraće se i `off()` metoda, čime će dalje slušanje događaja biti otkazano.

Metoda one() za pretplatu na događaje

Efekat identičan onom iz upravo prikazanog primera može se postići i korišćenjem samo jedne metode – `one()`. Metoda `one()` omogućava da se prilikom pojave nekog događaja pripadajuća logika izvrši samo jednom:

```
$("#my-button").one("click", function () {
    alert("The element was clicked.");
});
```

Umesto metode `on()`, za pretplatu na događaj je sada iskorišćena metoda `one()`. Na taj način će definisana anonimna funkcija biti izvršena samo jednom; odnosno, nakon obrade prvog događaja, automatski će biti obavljena odjava.

Rezime

- Selektovanje elemenata DOM strukture korišćenjem jQueryja obavlja se upotrebom jedne od specijalnih funkcija `$()` ili `jQuery()`.
- Funkcije za selektovanje DOM elemenata prihvataju jedan parametar kojim se definiše kriterijum na osnovu koga će biti obavljeno selektovanje jednog ili više elemenata.
- Prilikom definisanja parametra koji se prosleđuje jQuery funkcijama za selektovanje primenjuju se pravila CSS jezika za definisanje selektora.

- Za dobijanje roditelja nekog elementa moguće je koristiti metode `parent()`, `parents()` i `parentsUntil()`.
- Za pretragu elemenata naslednika, odnosno dece, jQuery obezbeđuje metode `children()` i `find()`.
- Metode koje se mogu koristiti za pretragu srodnika su `siblings()`, `next()`, `nextAll()`, `nextUntil()`, `prev()`, `prevAll()` i `prevUntil()`.
- Kada selektovanje podrazumeva veći broj elemenata, moguće je koristiti jQuery metode za filtriranje dobijenih elemenata – `first()`, `last()`, `eq()`, `filter()` i `not()`.
- jQuery poseduje veliki broj metoda čija imena odgovaraju događajima za koje se želi izvršiti pretplata.
- Registrovanje logike koja će se aktivirati prilikom pojave događaja korišćenjem jQuery biblioteke se veoma lako može postići i korišćenjem metode `on()`.
- Metoda `one()` omogućava da se prilikom pojave nekog događaja pripadajuća logika izvrši samo jednom.