Rukovanje sadržajem, strukturom i stilizacijom pomoću jQuery biblioteke

Prethodne lekcije donele su priču o različitim načinima za selektovanje elemenata i obradu događaja korišćenjem jQuery biblioteke. Selektori omogućavaju dobijanje, pretragu i filtriranje elemenata, a pretplata na događaje definisanje logike koja će se izvršiti u tačno definisanom trenutku.

Sve su to preduslovi kako bi bilo moguće preduzeti neke naredne korake u manipulaciji DOM strukturom. Stoga će lekcija pred vama biti posvećena različitim tehnikama koje omogućavaju da se elementi DOM strukture dodatno obrade nakon njihovog selektovanja.

Kompletna materija ove lekcije biće podeljena na sledeće tri logičke celine:

- manipulacija sadržajem
- manipulacija stilizacijom
- manipulacija čvorovima DOM strukture

Rukovanje sadržajem elemenata

Rukovanje sadržajem podrazumeva čitanje i definisanje tekstualnog sadržaja HTML elemenata. Takvi zahvati za nas nisu potpuna nepoznanica, s obzirom na to da su u prethodnim lekcijama već iskorišćeni neki osnovni pristupi za čitanje sadržaja elemenata.

Za rukovanje sadržajem elemenata jQuery poseduje tri metode, prikazane tabelom 12.1.

Metoda	Opis
`text()`	čita ili postavlja tekstualni sadržaj elementa
`html()`	čita ili postavlja tekstualni sadržaj elementa zajedno sa HTML kodom (ukoliko postoji)
`val()`	čita ili postavlja vrednosti polja HTML formi

Tabela 12.1. jQuery metode za rukovanje sadržajem

Tri prikazane metode koriste se za obavljanje sličnog posla. Ipak, svaka od metoda iz tabele 12.1. poseduje svoje karakteristične osobine i mesta primene. Prilikom odabira metode koja će biti korišćena, prvo je potrebno znati tip elementa čijim sadržajem se želi rukovati. Ukoliko je potrebno manipulisati sadržajem `input` elemenata, koristi se metoda `val()`. Za sve ostale elemente, potrebno je koristiti metode `text()` i `html()`. Ipak, i između ove dve metode postoje izvesne razlike. Sve osobine prikazanih metoda biće detaljno objašnjene u nastavku. Prvo će biti ilustrovane osobine `text()` i `html()` metoda kada se one koriste za čitanje sadržaja.

Čitanje sadržaja metodama text() i html()

Za testiranje metoda `text()` i `html()`, prvo će biti upotrebljen sledeći HTML element:

```
<h1 id="heading1"> This is heading 1</h1>
```

Kada se koristi metoda `text()`, čitanje izgleda ovako:

```
let h1Text = $("#heading1").text();
console.log(h1Text);
```

Na ovaj način, unutar konzole se dobija:

```
This is heading 1
```

Identičan efekat se dobija i kada se za čitanje u ovoj situaciji koristi metoda `html()`:

```
let h1Text = $("#heading1").html();
console.log(h1Text);
```

Ipak, kako ne biste pomislili da metode `text()` i `html()` imaju identičan efekat, biće prikazan još jedan primer:

```
<ul id="my-list">
    <li>one</li>
    <li>two</li>
    <li>three</li>
</ul>
```

Sada će metode `text()` i `html()` biti upotrebljene za čitanje sadržaja prikazane neuređene liste. Prvo metoda `text()`:

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

Na ovaj način, unutar konzole se dobija:

```
one
two
three
```

Metodom `text()` u prikazanom primeru se obavlja čitanje sadržaja neuređene liste. S obzirom na to da `ul` element nema direktan tekstualni sadržaj, metoda `text()` obavlja čitanje tekstualnog sadržaja stavki takve neuređene liste.

Kada se u identičnoj situaciji upotrebi metoda `html()`, rezultat je drugačiji:

```
let ulHtml = $("#my-list").html();
console.log(ulHtml);
```

Sada se unutar konzole dobija:

```
<li>one</li>
<li>two</li>
<li>three</li>
```

Poređenjem dobijenih ispisa dva prikazana primera, može se uvideti osnovna razlika između metoda `text()` i `html()`. Metoda `html()` čita i HTML kod koji se nalazi unutar nekog elementa, a metoda `text()` samo tekst.

Definisanje sadržaja metodama text() i html()

Slične su osobine metoda `text()` i `html()` kada se one koriste za postavljanje tekstualnog sadržaja elemenata. Kao referentni element za demonstraciju poslužiće:

```
<h1 id="heading1"> This is heading 1</h1>
```

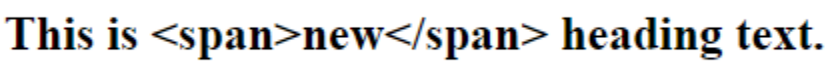
Tekst ovog naslova se može promeniti na sledeći način:

```
$("#heading1").text("This is new heading text.");
```

Ovo je osnovni primer postavljanja tekstualnog sadržaja elementa korišćenjem `text()` metode. Njoj je prosleđen tekst koji će biti postavljen unutar `h1` elementa. Ipak, ukoliko se metodi `text()` prosledi tekst koji sadrži neke HTML elemente, takvi elementi neće biti tretirani kao HTML kod, već kao običan tekst:

```
$("#heading1").text("This is <span>new</span> heading text.");
```

Sada je reč *new* smeštena unutar zasebnog `span` elementa. Ipak, izgled naslova na stranici nakon izvršavanja prikazane linije koda će biti kao na slici 12.1.

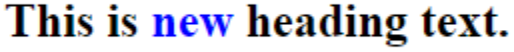


Slika 12.1. Sadržaj h1 elementa kreiran text() metodom

Ukoliko se umesto metode `text()` upotrebi metoda `html()`, stvari su drugačije:

```
$("#heading1").html("This is <span style='color:blue;'>new</span>
heading text.");
```

Sada je za postavljanje sadržaja elementa upotrebljena metoda `html()`. Takođe, i tekst koji se metodi prosleđuje je blago modifikovan, dodavanjem određene linijske stilizacije na `span` element. Efekat unutar web pregledača će biti kao na slici 12.2.



Slika 12.2. Sadržaj h1 elementa kreiran html() metodom

Na osnovu prikazanih primera se jasno može zaključiti da, prilikom postavljanja sadržaja elemenata, metoda `html()` specifične HTML elemente tretira kao HTML kod, dok to nije slučaj sa metodom `text()`.

Metoda val() za čitanje i postavljanje sadržaja

Metode `text()` i `html()` nije moguće koristiti za manipulaciju sadržajem svih HTML elemenata. Ukoliko je potrebno manipulisati sadržajem `input`, `select` i `textarea` elemenata, neophodno je koristiti metodu `val()`. Pri tome se pod `input` elementima misli na one elemente koji su namenjeni unosu tekstualnog sadržaja, odnosno elemente tipa `text`, `email`, `password`...

Postavljanje vrednosti `input` elementa tipa `text` može da izgleda ovako:

```
$("#name").val("default name");
```

Vrednost `input` elementa se može pročitati na sledeći način:

```
let nameValue = $("#name").val();
```

Čitanje vrednosti selektovane radio opcije

Metoda `val()` se može koristiti i za čitanje vrednosti `input` elemenata tipa `radio`. Ipak, ono što nas najviše zanima kada je reč o `radio` elementima jeste dobijanje vrednosti `radio` elementa koji je selektovan. To se može postići kombinovanjem metode `val()` i jednog specifičnog selektora:

```
        <form id="my-form">
            <input id="male" type="radio" name="gender" value="male"> <label
for="male" checked>Male</label>
            <input  id="female"  type="radio"  name="gender"  value="female">
<label for="female">Female</label>
            <input type="submit" value="Submit">
        </form>


        <script>
            $("#my-form").submit(function (e) {

                e.preventDefault();

                let selectedGender = $("input[name='gender']:checked").val();
                alert(selectedGender);

            });
        </script>
```

Primer prikazuje HTML dokument sa jednom formom. Unutar forme se nalazi grupa `radio` elemenata, koja korisnicima omogućava odabir pola. Unutar `script` elementa je definisan kod koji se aktivira kada se forma prosledi. Tada se čita i vrednost `radio` elementa koji je korisnik odabrao. To se postiže korišćenjem jednog složenog jQuery selektora – selektuje se `input` element sa `gender` vrednošću `name` atributa, ali samo onaj `input` element koji je korisnik odabrao. Takav element se dobija navođenjem pseudoklase `:checked`. Na kraju, vrednost selektovanog `radio` elementa se dobija korišćenjem metode `val()`. Stoga, kada korisnik klikne na *Submit* dugme, u modalnom prozoru će se prikazati vrednosti odabranog `radio` elementa.

Čitanje vrednosti input elementa tipa checkbox

Metoda `val()` se može koristiti i za čitanje vrednosti `input` elemenata tipa `checkbox`. Ipak, baš kao i kod `input` elemenata tipa `radio`, sama vrednost `checkbox` elementa za nas nema posebnu važnost. Ono što je mnogo značajnije jeste to da li je `checkbox` čekiran ili ne. Takva provera se može obaviti na sledeći način:

```
        <form id="my-form">
            <input      type="checkbox"      name="newsletter"      value="yes"
id="newsletter">
            <label for="newsletter">I would like to receive newsletter.</label>

            <input type="submit" value="Submit">
        </form>


        <script>
            $("#my-form").submit(function (e) {

                e.preventDefault();

                let newsletterChecked = $("#newsletter").is(":checked");
                alert(newsletterChecked);

            });
        </script>
```

Kod ilustruje HTML dokument sa formom unutar koje se nalazi jedan `input` element tipa `checkbox`. Unutar `script` elementa definisan je kod za pretplatu na događaj prosleđivanja forme. Kada se forma prosledi, obavlja se utvrđivanje toga da li je checkbox čekiran ili ne. Metoda kojom se tako nešto utvrđuje jeste `is()`. Reč je o metodi kojom se proverava zadovoljenje nekog uslova. S obzirom na to da je ovoj metodi u primeru prosleđena pseudoklasa `:checked`, na ovaj način se utvrđuje da li je checkbox čekiran ili ne. Metoda `is()` će kao svoj rezultat vratiti `true` ili `false` vrednost.

Pitanje

Ukoliko je unutar nekog HTML elementa potrebno postaviti jednostavan tekst, bez HTML koda, najbolje je koristiti metodu:

- **text()**
- value()
- html()
- val()

Objašnjenje:

Ukoliko je unutar HTML elementa potrebno postaviti jednostavan tekst, bez HTML koda, najbolje je koristiti metodu text().

Rukovanje stilizacijom

Osnovni način za rukovanje stilizacijom elemenata korišćenjem jQueryja podrazumeva korišćenje metode `css()` (tabela 12.2).

Metoda	Opis
`css()`	jQuery metoda za čitanje i postavljanje vrednosti CSS svojstava

Tabela 12.2. Osnovna jQuery metoda za rukovanje stilizacijom

Kada se metoda `css()` koristi za čitanje osobina stilizacije nekog elementa, njoj se prosleđuje naziv CSS svojstva čiju vrednost očekujemo:

```
let textSize = $("#heading1").css("font-size");
console.log("h1 text size is: " + textSize);
```

Bitno je razumeti da metoda `css()` omogućava čitanje takozvane proračunate (*computed*) stilizacije. Reč je o stilizaciji koja zavisi od brojnih faktora – CSS opisa, roditeljske stilizacije, podrazumevane stilizacije web pregledača. Neki element za određena CSS svojstva može imati proračunatu stilizaciju, pa čak i ako je mi samostalno nismo definisali. Takva je situacija i u prikazanom primeru. Čak i ukoliko se veličina teksta ne navede eksplicitno, unutar konzole se dobija ispis:

```
h1 text size is: 24px
```

Odakle je došla ova vrednost od 24px? Reč je o proračunatoj vrednosti koja dolazi od podrazumevane stilizacije web pregledača. Naime, web pregledači podrazumevano postavljaju veličinu `h1` naslova na 1.5em. Ukoliko se zna da je podrazumevana veličina standardnog teksta 16px, veoma lako se dolazi do vrednosti od 24px = 16px * 1,5.

Kada se metoda CSS koristi za postavljanje neke stilizacije, ona se koristi u sledećem obliku:

```
$("#heading1").css("font-size", "32px");
```

Kada se koristi za definisanje stilizacije, metoda `css()` prihvata dva parametra – prvi se odnosi na naziv CSS svojstva, a drugi na njegovu vrednost.

Neophodno je razumeti da jQuery omogućava definisanje linijske stilizacije. Drugim rečima, definisanjem stilizacije korišćenjem `css()` metode bivaju kreirani CSS opisi koji se nalaze na samom HTML elementu, kao deo vrednosti `style` svojstva. Tako struktura `h1` elementa u HTML kodu nakon prikazane intervencije izgleda ovako:

```
<h1 id="heading1" style="font-size: 32px;"> This is heading 1</h1>
```

Rukovanje stilizacijom dodavanjem i uklanjanjem klasa i korišćenjem metoda show() i hide()

Pored direktnog postavljanja linijske stilizacije korišćenjem metode `css()`, veoma česta praksa jeste rukovanje stilizacijom dodavanjem ili uklanjanjem CSS klasa. Za obavljanje takvog posla, jQuery poznaje metode ilustrovane tabelom 12.3.

Metoda	Opis
`addClass()`	dodaje jednu ili više klasa na selektovani element
`removeClass()`	uklanja jednu ili više klasa sa selektovanog elementa
`toggleClass()`	naizmenično uklanja postojeću klasu sa selektovanog elementa ili dodaje nepostojeću klasu na selektovani element
`hasClass()`	proverava da li selektovani element poseduje neku klasu

Tabela 12.3. jQuery metode za rukovanje klasama na elementima

Ukoliko je potrebno manipulisati isključivo vidljivošću nekog HTML elementa, moguće je koristiti jQuery metode prikazane tabelom 12.4.

Metoda	Opis
`show()`	prikazuje HTML element tako što postavlja vrednost njegovog CSS `display` svojstva na podrazumevanu vrednost (`block`, `inline`...)
`hide()`	skriva HTML element tako što postavlja vrednost njegovog CSS `display` svojstva na `none`
`toggle()`	naizmenično menja vidljivost elementa tako što element koji se vidi skriva, a skriveni element prikazuje

Tabela 12.4. jQuery metode za prikazivanje i skrivanje elemenata

U nastavku će biti prikazano nekoliko različitih načina za realizaciju jednog istog primera. Primeri će podrazumevati upotrebu metoda prikazanih tabelama 12.3 i 12.4.

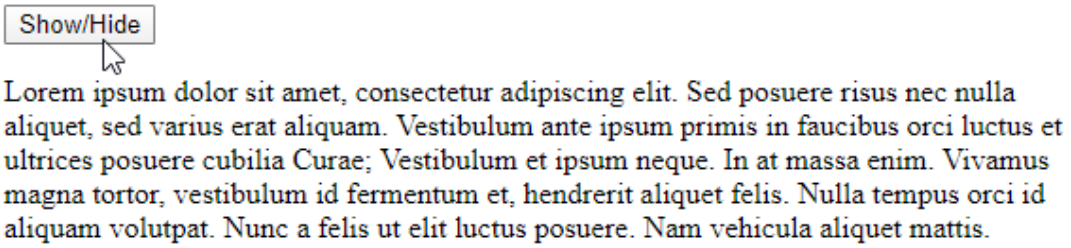
Za realizaciju primera biće iskorišćena sledeća stilizacija:

```
.shown {
    display: block;
}

.hidden {
    display: none;
}
```

HTML struktura će izgledati ovako:

```
<button id="my-button">Show/Hide</button>

<p id="my-paragraph">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed posuere risus nec nulla aliquet, sed varius erat aliquam. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Vestibulum et ipsum neque. In at massa enim. Vivamus magna tortor, vestibulum id fermentum et, hendrerit aliquet felis. Nulla tempus orci id aliquam volutpat. Nunc a felis ut elit luctus posuere. Nam vehicula aliquet mattis.</p>
```

Dokument će na stranici izgledati kao na slici 12.3.



Slika 12.3. Izgled HTML dokumenta u pregledaču

U nastavku će biti prikazano nekoliko različitih načina kojima se može postići naizmenično skrivanje i prikazivanje paragrafa, klikom na `button` element.

Pristup 1 – addClass(), removeClass(), hasClass()

```
let myParagraph = $("#my-paragraph");
myParagraph.addClass("shown");

$("#my-button").click(function () {

    if (myParagraph.hasClass("shown")) {
        myParagraph.removeClass("shown");
        myParagraph.addClass("hidden");
    } else {
        myParagraph.removeClass("hidden");
        myParagraph.addClass("shown");
    }
});
```

Prvi pristup za postizanje opisanog ponašanja podrazumeva korišćenje CSS klasa `.hidden` i `.shown`. Na početku se selektuje paragraf element i na njega se postavlja klasa `.shown` – zato

što je paragraf inicijalno vidljiv, a kako bi dalja logika funkcionisala ispravno. Nakon ove prve dve naredbe, obavljena je pretplata na `click` događaj `#my-button` elementa. Kada korisnik klikne na `#my-button` element, prvo se proverava da li na paragrafu postoji klasa `.shown`. Provera se postiže jQuery metodom `hasClass()`. Ukoliko klasa `.shown` postoji na paragrafu, ova metoda vraća vrednost `true`, pa izvršavanje ulazi u if blok. Unutar if bloka se `.shown` klasa uklanja sa elementa korišćenjem metode `removeClass()` i korišćenjem metode `addClass()` se obavlja postavljanje klase `.hidden`. Kada korisnik opet izvrši klik na `#my-button` element, biće aktiviran else uslovni blok, pa će biti obavljena obrnuta logika – uklanjanje klase `.hidden`, a dodavanje klase `.shown`.

Pristup 2 – toggleClass()

```
let myParagraph = $("#my-paragraph");

$("#my-button").click(function () {

    myParagraph.toggleClass("hidden");

});
```

Drugi pristup podrazumeva korišćenje samo jedne jQuery metode za rad sa klasama – `toggleClass()`. Takođe, sada se koristi i samo jedna CSS klasa – `.hidden`. Svakim klikom na `#my-button` element, paragrafu se dodaje ili uklanja klasa `.hidden`. Ukoliko element ne poseduje ovu klasu, što će biti slučaj prilikom učitavanja stranice, metoda `toggleClass()` će nju postaviti na paragraf i time će element postati nevidljiv. Sledećim klikom, metoda `toggleClass()` će klasu `.hidden` ukloniti sa paragrafa, pa će on ponovno postati vidljiv.

Pristup 3 – show(), hide()

```
let myParagraph = $("#my-paragraph");

$("#my-button").click(function () {

    if(myParagraph.css('display') == 'none'){
        myParagraph.show();
    } else {
        myParagraph.hide();
    }

});
```

Treći pristup za realizaciju primera podrazumeva korišćenje jQuery metoda `show()` i `hide()`. Ipak, kako bi se utvrdilo da li je element vidljiv ili ne, u primeru je iskorišćen i uslov unutar koga se proverava da li element poseduje svojstvo `display` sa vrednošću `none`. To je upravo vrednost koju metoda `hide()` postavlja za `display` svojstvo. Tako će se prilikom prvog klika na `#my-button` element aktivirati else uslovni blok, s obzirom na to da je paragraf inicijalno vidljiv, odnosno da poseduje `block` vrednost `display` svojstva. Biće pozvana `hide()` metoda, koja će vrednost `display` svojstva postaviti na `none`. Sledećim klikom na `#my-button` element, aktivira se if uslovni blok, pa paragraf ponovno postaje vidljiv.

Pristup 4 – toggle()

```
let myParagraph = $("#my-paragraph");

$("#my-button").click(function () {

    myParagraph.toggle();

});
```

Četvrta varijanta rešavanja prikazanog primera podrazumeva korišćenje `toggle()` jQuery metode. Reč je o metodi koja naizmenično menja vidljivost elementa. Tako je `toggle()` metoda zapravo zamena za metode `show()` i `hide()`. Ukoliko je element vidljiv, `toggle()` će učiniti da on postane nevidljiv, postavljanjem vrednosti `display` svojstva na `none`. Kada element ima vrednost `none`, `display` svojstva, metoda `toggle()` čini da on ponovno postane vidljiv, postavljanjem `display` svojstva na podrazumevanu vrednost u zavisnosti od tipa elementa.

Rukovanje čvorovima DOM strukture

U jednoj od prethodnih lekcija u kojima je bilo reči o objektnom modelu dokumenta, prikazani su različiti pristupi koji omogućavaju dinamičku izmenu DOM strukture korišćenjem osnovnih JavaScript pristupa. Pod pojmom dinamičke izmene DOM strukture podrazumeva se kreiranje novih čvorova (*nodes*) i izmena i brisanje postojećih. jQuery biblioteka olakšava i upravo spomenuti aspekt JavaScript programiranja. Olakšica dolazi u vidi nekoliko metoda, ilustrovanih tabelom 12.5.

Metoda	Opis
`append()`	dodaje sadržaj na kraj selektovanog elementa
`prepend()`	dodaje sadržaj na početak selektovanog elementa
`after()`	dodaje sadržaj nakon elementa nad kojim se pozove
`before()`	dodaje sadržaj pre elementa nad kojim se pozove
`remove()`	uklanja selektovani element i sve njegove potomke
`empty()`	uklanja kompletan sadržaj elementa, uključujući i sve njegove potomke

Tabela 12.5. jQuery metode za kreiranje i brisanje DOM čvorova

Metode prikazane tabelom 12.5. mogu se podeliti u dve grupe:

```
metode za dodavanje sadržaja - append(), prepend(), after(), before()
metode za uklanjanje sadržaja - remove(), empty()
```

Metode za dodavanje sadržaja

Metode za dodavanje sadržaja mogu se koristiti za dodavanje običnog tekstualnog sadržaja elementima:

```
<p id="my-paragraph">This is paragraph original text.</p>

<script>
    $("#my-paragraph").append('*new text');
</script>
```

Nakon izvršavanja ovakvog koda, `#my-paragraph` će imati sledeći sadržaj:

```
This is paragraph original text.*new text
```

Jasno je da je tekst prosleđen metodi `append()` dodat na kraj paragraf elementa.

Ipak, prava moć jQuery metoda za dodavanje sadržaja ogleda se u mogućnosti dodavanja novih HTML elemenata u DOM strukturu. Da bi se tako nešto postiglo, prvo je potrebno obaviti kreiranje takvih elemenata. Kreiranje novih HTML elemenata se može postići korišćenjem osnovnih jQuery funkcija za selektovanje:

```
let newParagraph = $("<p>This is dynamically created paragraph.</p>");
```

Prikazanom naredbom obavljeno je kreiranje jednog HTML elementa, preciznije paragrafa. HTML kod takvog elementa definisan je u `string` obliku i on je prosleđen osnovnoj jQuery funkciji. Na taj način je dobijen jedan jQuery objekat koji je dodeljen promenljivoj `newParagraph`.

Ovako kreiran element postoji samo interno, unutar programske JavaScript logike. Drugim rečima, on još nije vidljiv na stranici zato što nije uključen u DOM strukturu. Da bi se uključio u DOM strukturu, potrebno je iskoristiti neku od metoda za dodavanje sadržaja koje su prikazane tabelom 12.5.

```
$("body").append(newParagraph);
```

Ovom naredbom, kreirani HTML element dodat je `body` elementu HTML dokumenta. U naredbi se poziva jQuery metoda `append()`, čime se ovakav element dodaje na kraj `body` dela. Tekst stranice u web pregledaču nakon ove intervencije biće ovakav:

```
This is original, HTML paragraph.
This is dynamically created paragraph.
```

Ukoliko je novi HTML element potrebno dodati na početak nekog elementa, koristi se metoda `prepend()`:

```
let newParagraph = $("<p>This is dynamically created paragraph.</p>");
$("body").prepend(newParagraph);
```

Sada će tekst stranice unutar web pregledača biti ovakav:

```
This is dynamically created paragraph.
This is original, HTML paragraph.
```

Ukoliko je preciznije potrebno odrediti mesto na koje će biti postavljen novi HTML element, moguće je koristiti metode `after()` i `before()`:

```
<p id="my-paragraph">This is original HTML paragraph.</p>
<p id="my-paragraph2">This is another original HTML paragraph.</p>
<script>

    let newParagraph = $("<p>This is dynamically created paragraph.</p>");
    $("#my-paragraph").after(newParagraph);
</script>
```

Primer sada poseduje dva paragrafa koja izvorno postoje unutar HTML dokumenta. Definisanim JavaScript kodom je obavljeno kreiranje novog paragrafa, koji je metodom `after()` u strukturu dokumenta ubačen odmah nakon prvog paragrafa, zato što je metoda `after()` pozvana nad objektom koji predstavlja prvi paragraf. U web pregledaču će se dobiti:

```
This is original HTML paragraph.
This is dynamically created paragraph.
This is another original HTML paragraph.
```

Metode za uklanjanje sadržaja

Pored metoda za dodavanje, jQuery poseduje i metode za uklanjanje sadržaja i HTML elemenata. Metoda **remove()** uklanja element nad kojim se pozove:

```
<p id="my-paragraph">This is original HTML paragraph.</p>


<script>
    $("#my-paragraph").remove();
</script>
```

Primer podrazumeva postojanje jednog paragrafa. Unutar `script` elementa, takav paragraf se selektuje i nad dobijenim jQuery objektom se poziva metoda `remove()`. Na taj način se paragraf `#my-paragraph` potpuno uklanja iz DOM strukture. Bitno je znati da metoda `remove()` uklanja i sve eventualne potomke elementa nad kojim se pozove.

Ukoliko je potrebno ukloniti kompletan sadržaj nekog elementa, uključujući i sve njegove potomke, ali ne i sam element, može se koristiti metoda **empty()**:

```
<p id="my-paragraph">This is original HTML paragraph.</p>

<script>
    $("#my-paragraph").empty();
</script>
```

Sada je na identičnom primeru iskorišćena metoda `empty()`. Efekat unutar web pregledača je identičan, ali ukoliko se pogleda kod HTML stranice, može se videti da `#my-paragraph` element i dalje postoji:

```
<p id="my-paragraph"></p>
```

Element `my-paragraph` i dalje je prisutan unutar stranice. Metoda `empty()` je uklonila samo njegov sadržaj.

Rezime

- Metoda `text()` čita ili postavlja tekstualni sadržaj elementa.
- Metoda `html()` čita ili postavlja tekstualni sadržaj elementa zajedno sa HTML kodom.
- Metoda `val()` čita ili postavlja vrednosti polja HTML formi.
- Osnovni način za rukovanje stilizacijom elemenata korišćenjem jQueryja podrazumeva korišćenje metode `css()`.
- Pored direktnog postavljanja linijske stilizacije korišćenjem metode `css()`, veoma česta praksa jeste rukovanje stilizacijom dodavanjem ili uklanjanjem CSS klasa.
- Metoda `addClass()` dodaje jednu ili više klasa na selektovani element.
- Metoda `removeClass()` uklanja jednu ili više klasa sa selektovanog elementa.
- Metoda `toggleClass()` naizmenično uklanja postojeću klasu sa selektovanog elementa ili dodaje nepostojeću klasu na selektovani element.
- Metoda `hasClass()` proverava da li selektovani element poseduje neku klasu.
- Metoda `show()` prikazuje HTML element.
- Metoda `hide()` skriva HTML element.
- Metoda `toggle()` naizmenično menja vidljivost elementa.
- jQuery metode za dodavanje sadržaja ili elemenata su `append()`, `prepend()`, `after()` i `before()`.
- jQuery metode za uklanjanje sadržaja ili elemenata su `remove()` i `empty()`.