XML i JavaScript

Korišćenjem XML-a, procesom markiranja podataka dobija se prilično razumljiv, lako čitljiv kod. Osobina lake čitljivosti svakako je prilično korisna kada je potrebno upoznati se sa osnovnom strukturom dokumenta ili pronaći neki podatak jednostavnim uvidom u tekstualni sadržaj XML fajla. Ipak, bitno je razumeti da je XML prevshodno namenjen jednoj posebnoj grupi čitalaca – XML parserima. Reč je o softverskim bibliotekama, odnosno funkcionalnostima, čiji je posao da čitanju i razumeju XML kod.

Pored čitanja, i stvaranje XML koda veoma retko se obavlja ručno. I takav posao sprovode posebne softverske komponente – serijalizatori, koji su zaduženi da podatke kojima aplikacije barataju tokom njihovog izvršavanja pretvore u XML kod.

S obzirom na to da je predmet našeg interesovanja frontend programiranje, u lekciji koja je pred vama biće prikazani pristupi koji omogućavaju da se XML kreira i pročita korišćenjem jezika JavaScript.

Parsiranje, serijalizacija i deserijalizacija

Pre nego što se upustimo u praktičan rad koji se tiče čitanja i kreiranja XML koda, bitno je razumeti nekoliko pojmova koji će se koristiti u redovima koji slede. Tako se prilikom rada sa podacima veoma često mogu čuti pojmovi parsiranje, serijalizacija i deserijalizacija.

Parsiranje je pojam koji se odnosi na čitanje nekog teksta i njegovu obradu u cilju izvlačenja upotrebljivih podataka. S obzirom na to da je XML kod običan tekst unutar koga se za obeležavanje podataka koriste različite oznake, pojam parsiranja veoma često se koristi prilikom rada sa XML-om. Tako je parsiranje XML-a zapravo čitanje XML koda i izvlačenje podataka koji su takvim kodom markirani.

Pojmovi serijalizacije i deserijalizacije takođe se mogu čuti prilikom rada sa XML-om. Serijalizacija se odnosi na pojam konvertovanja objekata koji se nalaze unutar nekog programa u oblik koji se lako može sačuvati ili proslediti. Kako bi se podaci koji su serijalizovani vratili u svoj izvorni, objektni oblik, vrši se proces deserijalizacije. Serijalizacija i deserijalizacija često se drugačije nazivaju *marshalling* i *unmarshalling*, respektivno.

JavaScript jezik ne poseduje ugrađene mehanizme za obavljanje parsiranja, odnosno serijalizacije XML jezika. Ipak, web pregledači u skupu funkcionalnosti koje izlažu na korišćenje poseduju nekoliko API-a koje je moguće koristiti za programabilno rukovanje XML-om na klijentskom delu web aplikacija.

Parsiranje XML-a korišćenjem DOMParser-a

Priča o manipulaciji XML-om korišćenjem JavaScript jezika započinje osnovnim načinom za parsiranje XML koda. Za obavljanje takvog posla, web pregledači izlažu jedan poseban skup funkcionalnosti, odnosno aplikativni programski interfejs **DOMParser**.

Prvi primer korišćenja DOMParsera, podrazumevaće parsiranje veoma jednostavnog XML koda:

```
var myXmlString = '<root><message>Hello World!</message></root>';
```

XML je sačinjen iz korenog elementa `root`, unutar koga se nalazi jedan element `message` sa sadržajem `Hello World!`.

Pristup DOMParser funkcionalnostima započinje kreiranjem objekta za parsiranje:

```
var domParser = new DOMParser();
```

Objekat za parsiranje, čija referenca je prikazanom naredbom smeštena unutar promenljive `domParser`, poseduje jednu metodu – `parseFromString()`.

Metoda parseFromString()

Metoda `parseFromString()`, `DOMParser` objekta koristi se za parsiranje XML teksta, odnosno za konvertovanje XML teksta u DOM. Njena sintaksa izgleda ovako:

```
parseFromString(string, mimeType)
```

Metoda `parseFromString()` prihvata dva parametra:

- `string` – string koji sadrži XML, HTML ili SVG kod; prosleđeni sadržaj ovog parametra biće parsiran
- `mimeType` – vrednost kojom se parseru govori koji jezik za markiranje da očekuje unutar teksta prvog parametra; od vrednosti ovoga parametra zavisi i povratna vrednost metode `parseFromString()`

Tri osnovne vrednosti koje se metodi `parseFromString()`mogu proslediti kao drugi parametar prikazane su tabelom 3.1.

mimeType	Povratna vrednost
`text/html`	`HTMLDocument`
`text/xml`	`XMLDocument`
`image/svg+xml`	`SVGDocument`

Tabela 3.1. Vrednosti mimeType parametra parseFromString() metode

U tabeli 3.1. se može videti na koji način vrednost drugog parametra utiče na povratnu vrednost metode `parseFromString()`.

Na osnovnu upravo iznetih osobina metode `parseFromString()` može se zaključiti da DOMParser nije namenjen parsiranju isključivo XML koda, već se može koristiti i za parsiranje HTML-a i XML izvedenog jezika za opisivanje vektorske grafike – SVG. Ipak, mi ćemo u narednim redovima DOMParser koristiti samo za parsiranje XML-a.

Primer korišćenja DOMParser funkcionalnosti za parsiranje XML-a može da izgleda ovako:

```
var myXmlString = '<root><message>Hello World!</message></root>';
var domParser = new DOMParser();
var xmlDom = domParser.parseFromString(myXmlString, "text/xml");
```

Prvom naredbom je definisan već prikazani XML u string obliku, kao vrednost promenljive `myXmlString`. Zatim su obavljeni kreiranje parsera i poziv metode za parsiranje. Povratna vrednost metode za parsiranje smeštena je unutar promenljive `xmlDom`. Naziv ove promenljive nije odabran slučajno. Naime, parsiranjem XML-a korišćenjem DOMParsera dobija se objektna reprezentacija XML koda, predstavljena u formi XML DOM-a.

XML DOM

DOM je skraćenica koja označava pojam *Document Object Model.* DOM je objektna reprezentacija dokumenata koji se kreiraju nekim od jezika za obeležavanje. Stoga je DOM zapravo jezički nezavistan standard, koji programima i skriptama omogućava da pristupe sadržaju, strukturi i stilizaciji dokumenata. DOM se može koristiti prilikom rada sa HTML i XML dokumentima.

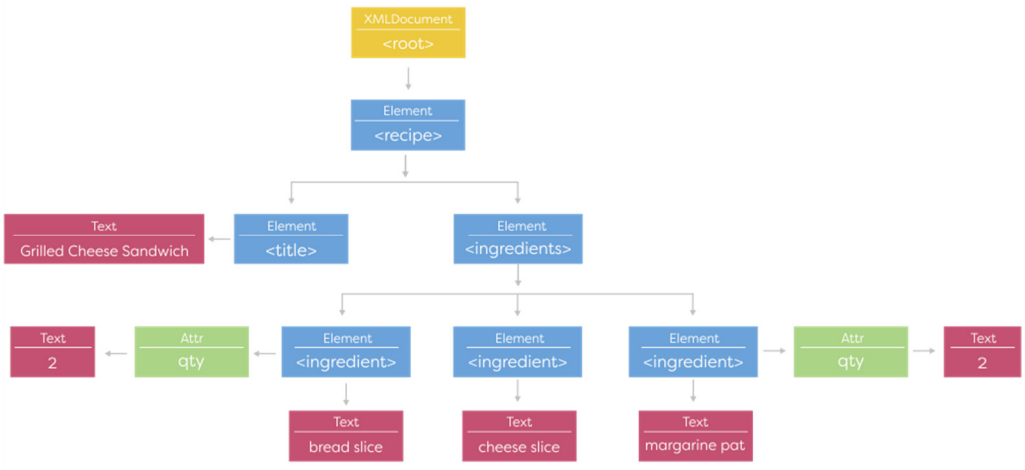
Kada se govori o HTML dokumentima, upotrebljava se pojam HTML DOM, a kada je reč o XML dokumentima, govori se o XML DOM-u. Ipak, reč je o objektnom modelu sa potpuno istim osobinama, pa ukoliko znate da koristite HTML DOM, bez problema ćete koristiti i XML DOM.

Jedina razlika se ogleda u postojanju određenog broja predefinisanih objekata u slučaju HTML DOM-a (`HTMLElement`, `HTMLBodyElement`, `HTMLTableElement`...), zbog činjenice da HTML poznaje unapred definisane elemente. Takvih unapred definisanih objekata u XML DOM-u nema, zato što tvorac dokumenta samostalno kreira elemente.

DOM je zapravo objekta struktura u formi stabla. Stablo je specijalna vrsta strukture podataka, kod koje svaki element, osim korenog, ima svog roditelja. Takođe, svaki element može imati i proizvoljan broj potomaka. To je upravo struktura koju XML elementi međusobno grade unutar jednog XML dokumenta:

```
<root>
    <recipe>
        <title>Grilled Cheese Sandwich</title>
        <ingredients>
            <ingredient qty="2">bread slice</ingredient>
            <ingredient>cheese slice</ingredient>
            <ingredient qty="2">margarine pat</ingredient>
        </ingredients>
    </recipe>
</root>
```

Prikazani kod za opisivanje jednog kulinarskog recepta nam je poznat iz jedne od prethodnih lekcija. Objektni model ovakvog dokumenta (DOM) izgleda kao na slici 3.1.



Slika 3.1. Objektni model XML dokumenta za prikaz recepta

Na slici 3.1. se može videti objektni model XML dokumenta za opisivanje recepta. Svaki kvadrat na slici 3.1. drugačije se naziva čvor (*node*). Svaki čvor modeluje se korišćenjem posebnog objekta XML DOM-a:

- `XMLDocument` – objekat kojim se predstavlja koreni element dokumenta
- `Element` – objekat kojim se predstavljaju XML elementi
- `Attr` – objekat kojim se predstavljaju XML atributi
- `Text` – objekat kojim se predstavlja sadržaj XML elemenata i atributa

Pored pobrojanih objekata koji se koriste prilikom predstavljanja upravo prikazanog XML dokumenta u objektnom obliku, XML DOM poseduje još neke objekte:

- `EntityReference` – objekat kojim se predstavljaju entiteti za referenciranje
- `CDATASection` – objekat za predstavljanje sekcija koje čitač neće parsirati
- `Comment` – objekat za predstavljanje komentara

Pozivanjem metode `parseFromString()`, DOMParser automatski na osnovu priloženog XML koda formira njegovu objektnu reprezentaciju izrađenu XML DOM-om. U našem primeru se referenca na takvu objektnu reprezentaciju smešta unutar promenljive `xmlDom`. Na nama je da dalje odlučimo šta želimo da uradimo sa takvom objektnom predstavom XML dokumenta.

Evo kako bi izgledao ispis tekstualne vrednosti koja je markirana jednim XML elementom:

```
var myXmlString = '<root><message>Hello World!</message></root>';
var domParser = new DOMParser();
var xmlDom = domParser.parseFromString(myXmlString, "text/xml");
var messageElement = xmlDom.getElementsByTagName("message")[0];
var messageText = messageElement.childNodes[0];
console.log(messageText.nodeValue);
```

Izvršavanjem prikazanog koda, unutar konzole se dobija vrednost smeštena unutar `message` elementa:

```
Hello World!
```

Do elementa `message` dolazi se korišćenjem metode `getElementsByTagName()`, koja se poziva nad DOM referencom dobijenom od DOMParsera.

Metoda getElementsByTagName()

Metoda `getElementsByTagName()` koristi se za dolaženje do objekata koji predstavljaju reference na XML elemente. Metodi se prosleđuje naziv elementa. Ova metoda kao svoju povratnu vrednost emituje niz objekata.

S obzirom na to da u našem primeru postoji samo jedan `message` element, njime smo pristupili navođenjem indeksa `[0]` odmah nad povratnom vrednošću metode `getElementsByTagName()`. Ostatak postupka dolaska do vrednosti `Hello World!` potvrđuje ono što je izrečeno o osobinama DOM strukture. Tekstualni sadržaj nalazi se unutar zasebnog objekta tipa `Text`, a do njega smo došli korišćenjem svojstva **childNodes**, koje čuva referencu na niz svih DOM objekata koji se nalaze unutar nekog drugog DOM objekta. I tu je obavljen pristup prvom elementu, navođenjem indeksa `[0]`. Na kraju, do konkretne tekstualne vrednosti došli smo korišćenjem svojstva **nodeValue**.

Informacije o XML DOM čvorovima

U upravo prikazanom primeru iskorišćeno je jedno svojstvo za dolazak do teksta XML čvora. Reč je o svojstvu `nodeValue` koje nam omogućava da dođemo do određene informacije o XML DOM čvoru. Takvih svojstava je nekoliko:

- `nodeName` – naziv čvora
- `nodeValue` – vrednost čvora
- `nodeType` – tip čvora

Tipovi čvorova izražavaju se celobrojnim numeričkim vrednostima, pa je tako:

- `1 – Element`
- `2 – Attribute`
- `3 – Text`
- `8 – Comment`
- `9 – Document`

U nekim situacijama DOMParser neće biti u mogućnosti da konstruiše objektnu reprezentaciju dokumenta. To se može dogoditi ukoliko XML dokument nije ispravno formiran, odnosno ukoliko ne zadovoljava sintaksna pravila jezika. Bitno je znati da u takvim situacijama DOMParser ne emituje izuzetak, već XML kod sačinjen iz `<parsererror>` elementa unutar koga je spakovana poruka greške. Evo primera u kome se pokušava obaviti parsiranje sintaksno netačnog XML koda:

- `var myXmlString = '<root><message><Hello World!</message></root>';`
- `var domParser = new DOMParser();`
- `var xmlDom = domParser.parseFromString(myXmlString, "text/xml");`

XML kod sada poseduje jednu sintaksnu grešku – unutar vrednosti `message` elementa upotrebljen je specijalni XML karakter `<`. S obzirom na to da u ovakvim situacijama DOMParser ne izbacuje izuzetak, neophodno je da sami kreiramo logiku koja će obraditi ovakve situacije:

```
var myXmlString = '<root><message><Hello World!</message></root>';
var domParser = new DOMParser();
var xmlDom = domParser.parseFromString(myXmlString, "text/xml");

if (xmlDom.getElementsByTagName("parsererror").length !== 0) {
      console.log("There is an error while parsing XML
document.");
} else {
      var messageElement =
xmlDom.getElementsByTagName("message")[0];
      var messageText = messageElement.childNodes[0];
      console.log(messageText.nodeValue);
}
```

Sada je primeru dodat i uslovni blok kojim se proverava da li XML DOM sadrži element `parsererror`. Ukoliko sadrži, to znači da je došlo do greške tokom parsiranja dokumenta, pa se unutar konzole prikazuje odgovarajuća poruka.

Serijalizacija XML DOM-a korišćenjem XMLSerializera

Prethodni redovi ilustrovali su pristup koji omogućava parsiranje XML koda koji se nalazi unutar neke `string` JavaScript promenljive. Vrlo često se može javiti potreba i za obavljanjem suprotne operacije – generisanja XML koda korišćenjem klijentske logike web aplikacija. Za obavljanje takvog posla koristi se jedan poseban Web API – **XMLSerializer**.

XMLSerializer je vrlo jednostavan za korišćenje. Dovoljno je kreirati objekat za serijalizaciju i pozvati metodu `serializeToString()`:

```
var xmlSerializer = new XMLSerializer();
var xmlString = xmlSerializer.serializeToString(xmlDocument);
```

Iz prikazanog koda možete videti da metoda prihvata jedan parametar (`xmlDocument`) koji mi još nigde nismo definisali. Reč je, zapravo, o objektnoj reprezentaciji XML dokumenta koji će biti serijalizovan.

Metoda serializeToString()

XMLSerializer objekat za serijalizaciju poseduje samo jednu metodu, `serializeToString()`. Njena sintaksa je sledeća:

```
serializeToString(node)
```

Metoda `serializeToString()` može da prihvati jedan ulazni parametar:

- `node` – ukazuje na XML DOM objekat koji predstavlja XML kod koji je potrebno serijalizovati

Metoda `serializeToString()`može da prihvati objekat tipa `Node` ili bilo koji objekat koji njega nasleđuje. Bitno je znati da je `Node` osnovni roditeljski objekat za sve specifične DOM objekte, od kojih su najznačajniji prikazani na slici 3.1. Tako metoda `serializeToString()` može da prihvati objekte tipa: `Document`, `XMLDocument`, `Element`, `Attr`...

Kao svoju povratnu vrednost metoda `serializeToString()` emituje tekst koji predstavlja XML kod kreiran na osnovu prosleđenog XML DOM objekta.

Na osnovu upravo iznetih osobina XMLSerializer API-a, lako je zaključiti da on omogućava da se kreira XML kod na osnovu objektne reprezentacije izražene korišćenjem XML DOM-a. Stoga je za korišćenje XMLSerializera neophodno konstruisati objektnu reprezentaciju XML-a koji želimo da dobijemo u tekstualnom obliku.

Kreiranje XML-a u obliku objektne reprezentacije koja postoji unutar interne memorije JavaScript aplikacije započinje definisanjem dokumenta sa korenim elementom. To se postiže na sledeći način:

```
var xmlDocument = document.implementation.createDocument(null,
"root");
```

Koriste se ugrađeni `Document` objekat i njegovo svojstvo `implementation`, koje čuva referencu na objekat `DOMImplementation`. Takav objekat poseduje metodu `createDocument()`.

Metoda createDocument()

Metodom `createDocument()`obavlja se kreiranje XML dokumenta:

```
createDocument(namespaceURI, qualifiedNameStr);
```

Metoda `createDocument()` poseduje dva obavezna parametra:

- `namespaceURI` – prostor imena dokumenta koji će biti kreiran; ukoliko dokument neće pripadati nijednom prostoru imena, prosleđuje se vrednost `null`
- `qualifiedNameStr` – naziv korenog elementa dokumenta koji će biti kreiran

Kao svoju povratnu vrednost, metoda `createDocument()` emituje objekat tipa `XMLDocument`.

Na osnovnu prikazane sintakse metode `createDocument()`, može se zaključiti da se ona u prikazanom primeru koristi za kreiranje objektne reprezentacije jednog XML dokumenta koji će kao svoj koreni element imati element `root`.

Nakon kreiranja `XMLDocument` objekta koji će predstavljati XML dokument, može se preći na kreiranje elemenata, sadržaja i atributa. U nastavku će biti ilustrovano kreiranje objektne reprezentacije XML dokumenta za predstavljanje kulinarskog recepta, koji smo već videli u dosadašnjem toku lekcije.

Proces kreiranja objektne strukture nekog XML dokumenta podrazumeva nekoliko koraka:

- kreiranje čvorova, odnosno elemenata, atributa, teksta...;
- povezivanje kreiranih čvorova, odnosno dodeljivanje sadržaja elementima i atributima i smeštanje elemenata unutar drugih elemenata, čime se kreira odgovarajuća objektna struktura;
- smeštanje svih kreiranih elemenata unutar korenog elementa `XMLDocument` objekta.

Kreiranje XML čvorova

Proces kreiranja XML objektne strukture započinje kreiranjem XML čvorova. Za obavljanje takvog posla može se koristiti nekoliko metoda:

- `createElement(elementName)` – metoda za kreiranje novog elementa
- `createAttribute(attributeName)` – metoda za kreiranje novog atributa
- `createTextNode(text)` – metoda za kreiranje novog tekstualnog čvora

Kreiranje objektne XML strukture našeg recepta započećemo od elemenata koji se u XML strukturi nalaze na dnu hijerarhije, odnosno od elemenata `ingredient`:

```
var ingredient1Element = xmlDocument.createElement("ingredient");
```

Na ovaj način je kreiran jedan objekat koji predstavlja element `ingredient`. Ipak, on u ovom trenutku postoji samo unutar objekta čija je referenca smeštena u promenljivu `ingredient1Element` i nema nikakav sadržaj niti atribute. Stoga je sledeći korak da kreiramo njegov sadržaj:

```
var ingredient1ElementText = xmlDocument.createTextNode("bread
slice");
```

Ovo je sada naredba kojom će biti kreiran jedan tekstualni XML čvor sa sadržajem `bread slice`. Ipak, ovakav tekstualni čvor ni na koji način nije pridružen nešto ranije kreiranom elementu. Kako bismo to uradili, potrebno je koristiti metode za dodavanje čvorova.

Dodavanje čvorova

Nakon kreiranja elemenata, atributa ili tekstualnih čvorova, njih je potrebno priključiti pripadajućim elementima. To se može obaviti korišćenjem nekoliko metoda:

- `appendChild(element)` – postavlja element na kraj nekog drugog elementa
- `insertBefore(newNode, existingNode)` – dodaje element definisan prvim parametrom pre nekog drugog elementa, definisanog drugim parametrom
- `setAttributeNode(attribute)` – dodaje unapred kreirani atribut nekom elementu
- `setAttribute(attribute, value)` – kreira atribut i dodaje ga nekom elementu
- `insertData(start, text)` – dodaje tekst unutar tekstualnog čvora, na poziciju definisanu prvim parametrom

Tekstualni čvor biće pridružen elementu na sledeći način:

```
var ingredient1Element = xmlDocument.createElement("ingredient");
var ingredient1ElementText = xmlDocument.createTextNode("bread
slice");
ingredient1Element.appendChild(ingredient1ElementText);
```

Kreiranom `ingredient` elementu atribut `qty` se može dodati na sledeći način:

```
var ingredient1Attribute = xmlDocument.createAttribute("qty");
ingredient1Attribute.nodeValue = "2";
ingredient1Element.setAttributeNode(ingredient1Attribute);
```

Prvo je obavljeno kreiranje objekta atributa korišćenjem metode `createAttribute()`. Vrednost atributa je postavljena svojstvom `nodeValue`. Na kraju, kreirani atribut je pridružen elementu korišćenjem metode `setAttributeNode()`.

Kompaktniji način za kreiranje i dodavanje atributa može se postići upotrebom metode `setAttribute()`:

```
ingredient1Element.setAttribute("qty", "2");
```

Metoda `setAttribute()` omogućava da se kreiranje atributa i njegove vrednosti obavi zajedno sa njegovim pridruživanjem odgovarajućem elementu.

Izmena i brisanje čvorova

Ukoliko se barata nekom već kreiranom XML DOM strukturom, korisne mogu biti i metode za izmenu i brisanje postojećih čvorova. To se može postići korišćenjem sledećih metoda:

- `removeChild(element)` – uklanja prosleđeni element koji postoji unutar elementa nad kojim se ova metoda poziva
- `replaceChild(newElement, existingElement)` – menja drugi prosleđeni element prvim i to unutar nekog element nad kojim se ova metoda pozove
- `removeAttribute(attributeName)` – uklanja atribut sa prosleđenim nazivom koji se nalazi na elementu nad kojim se ova metoda poziva
- `replaceData(offset, length, text)` – metoda za zamenu dela teksta unutar tekstualnog čvora; prvi parametar definiše početak teksta koji će biti zamenjen, a drugi parametar koliko će karaktera biti zamenjeno; treći parametar je novi tekst koji će biti dodat umesto dela postojećeg teksta

Pitanje

Pretvaranje XML koda u DOM strukturu obavlja se korišćenjem:

a) DOMParser API-a
b) XMLSerializer API-a

Objašnjenje:

Proces konvertovanja XML teksta u XML DOM naziva se parsiranje. Parsiranje XML koda moguće je obaviti korišćenjem web API-a koji se zove DOMParser.

Proces kreiranja kompletne objektne strukture XML dokumenta koji predstavlja naš recept podrazumeva korišćenje funkcionalnosti koje su ilustrovane u prethodnim redovima. Kompletan primer u kome se tako nešto realizuje prikazan je u nastavku.

Primer – Kreiranje objektne XML DOM reprezentacije jednog recepta

U prethodnim redovima prikazan je kod za kreiranje objektne predstave samo jednog dela XML dokumenta koji predstavlja recept. Stoga će u nastavku biti prikazan primer koji će podrazumevati kompletan kod za dinamičko, programabilno kreiranje XML-a, koji će na kraju biti serijalizovan i prikazan kao tekst unutar HTML dokumenta.

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <title>XML Serializer</title>
    </head>

    <body>
```

```
        <div id="xml">
        </div>

        <script>

            //create XMLDocument object with root element
            var xmlDocument =
document.implementation.createDocument(null, "root");

            //create recipe
            var recipeElement = xmlDocument.createElement("recipe");

            //create title element
            var titleElement = xmlDocument.createElement("title");
            var titleElementText =
xmlDocument.createTextNode("Grilled Cheese Sandwich");
            titleElement.appendChild(titleElementText);

            //add title to recipe
            recipeElement.appendChild(titleElement);


            //create ingredients
            var ingredientsElement =
xmlDocument.createElement("ingredients");


            //create first ingredient element
            var ingredient1Element =
xmlDocument.createElement("ingredient");
            var ingredient1ElementText =
xmlDocument.createTextNode("bread slice");
            ingredient1Element.appendChild(ingredient1ElementText);
            ingredient1Element.setAttribute("qty", "2");


            //create second ingredient element
            var ingredient2Element =
xmlDocument.createElement("ingredient");
            var ingredient2ElementText =
xmlDocument.createTextNode("cheese slice");
            ingredient2Element.appendChild(ingredient2ElementText);


            //create third ingredient element
            var ingredient3Element =
xmlDocument.createElement("ingredient");
            var ingredient3ElementText =
xmlDocument.createTextNode("margarine pat");
            ingredient3Element.appendChild(ingredient3ElementText);
            ingredient3Element.setAttribute("qty", "2");

            //add ingredient(s) to ingredients
            ingredientsElement.appendChild(ingredient1Element);
            ingredientsElement.appendChild(ingredient2Element);
            ingredientsElement.appendChild(ingredient3Element);
```

```
                //add ingredients to recipe
                recipeElement.appendChild(ingredientsElement);

                //add recipe to root
                xmlDocument.documentElement.appendChild(recipeElement);

                //create XMLSerializer to convert XML DOM to string
                var xmlSerializer = new XMLSerializer();
                var xmlString =
    xmlSerializer.serializeToString(xmlDocument);

                //show xml in div element on html page
                var xmlContainer = document.getElementById("xml");

    xmlContainer.appendChild(document.createTextNode(xmlString));

            </script>
        </body>
    </html>
```

Kod predstavlja kompletnu HTML stranicu sa JavaScript kodom koji će prvo stvoriti objektnu, DOM reprezentaciju XML dokumenta, a zatim takav DOM pretvoriti u tekstualni XML kod.

Za realizaciju prikazanog primera korišćeni su pristupi i metode koji su u ovoj lekciji predstavljeni nešto ranije. Jedina novina jeste način na koji se XML kod u tekstualnom obliku prikazuje unutar HTML dokumenta. Kako bi se XML kod zaista i prikazao kao tekst, u primeru je eksplicitno kreiran jedan tekstualni čvor sa sadržajem koji predstavlja XML kod, a zatim je takav tekstualni čvor dodat `div` elementu sa `id`-em `xml`. Da je obavljeno direktno dodavanje XML koda `div` elementu (na primer, korišćenjem `innerHtml` svojstva), pregledač bi parsirao XML kod kao da je reč o HTML-u. Dodavanjem XML koda unutar tekstualnog čvora osigurava se njegov prikaz u vidu običnog teksta.

Rezime

- XML parseri su softverske biblioteke, odnosno funkcionalnosti čiji je posao da čitaju i razumeju XML kod.
- Parsiranje je pojam koji se odnosi na čitanje nekog teksta i njegovu obradu radi izvlačenja upotrebljivih podataka.
- Serijalizacija se odnosi na koncept konvertovanja objekata koji se nalaze unutar nekog programa u oblik koji se lako može sačuvati ili proslediti.
- Kako bi se podaci koji su serijalizovani ponovo vratili u svoj izvorni, objektni oblik, vrši se proces deserijalizacije.
- Parsiranje XML koda moguće je obaviti korišćenjem web API-a koji se zove DOMParser.
- Metoda `parseFromString()` DOMParser objekta koristi se za parsiranje XML teksta, odnosno za konvertovanje XML teksta u DOM.
- DOM je skraćenica koja označava pojam Document Object Model, a u slučaju XML-a reč je o XML DOM-u.
- Serijalizacija XML DOM-a može se obaviti korišćenjem Web API-a `XMLSerializer`.