**Rad sa podacima**

**00:14:33**

Jedinica: 29 od 30

**+Rezime**

Svaka aplikacija, bez obzira na tip ili tehnologiju kojom je realizovana, rukuje izvesnim podacima. Može se reći da je rad sa podacima osnova svake aplikacije. Kod HTML5 aplikacija, podaci su smešteni unutar tagova HTML dokumenata. U zavisnosti od elemenata koji takve podatke nose, podaci imaju različito značenje i namenu.

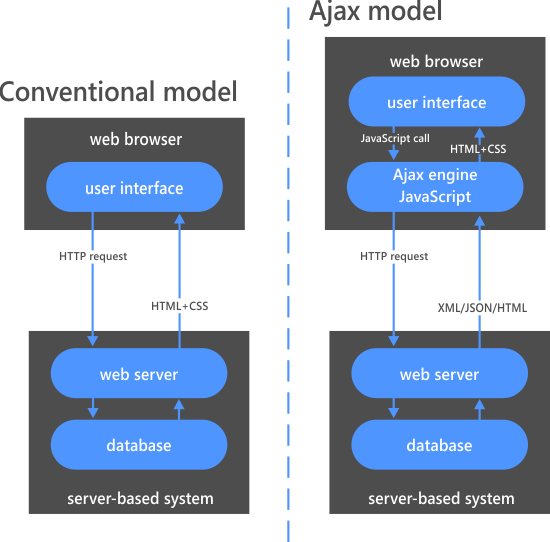
Rukovanje, odnosno rad sa podacima, uglavnom je zaduženje tehnologija koje postoje na [backendu](https://www.link-elearning.com/linkdl/opisPojma.php?id=144320). One su zadužene za čitanje, pisanje, brisanje i izmenu podataka, odnosno za direktnu komunikaciju sa skladištem. Tako u oblast kojom gospodari JavaScript podaci uglavnom već stižu formirani.

Opisan šablon za rad sa podacima predstavlja tradicionalni model web aplikacija.  Tradicionalne web aplikacije šalju i primaju informacije korišćenjem sinhronih zahteva:

* klijent uputi zahtev serveru za određenom stranicom
* server aktivira logiku definisanu korišćenjem neke od serverskih tehnologija, kako bi kreirao stranicu koju klijent zahteva
* nakon završetka serverske logike, korisniku se isporučuje stranica
* kada korisniku zatreba neka druga stranica ili informacija, postupak se ponavlja iz početka

JavaScript omogućava drugačiji pristup razmene informacija, tako što omogućava asinhrono slanje zahteva i prijem serverskog odgovora. Tako nije potrebno uvek iznova dobijati kompletnu stranicu, odnosno jedinica prenosa više nije stranica, već podatak. Tako klijent može da dobije samo podatke koji su mu u tom trenutku potrebni, bez osvežavanja kompletne stranice.

Uporedni prikaz tradicionalnog i asinhronog (AJAX) modela dat je na slici 29.1.



*Slika 29.1 – Tradicionalni i Ajax pristup*

**anica koja se koristi da označi skup tehnologija za rad sa podacima u asinhronom maniru korišćenjem JavaScripta je:**

 Ajax

 Flex

 Grudge

 Conventional

**Format prenosa podataka**

Dva najkorišćenija formata koja se koriste za prenos podataka asinhronim putem su: XML i JSON.

JSON je nestrukturirani format, koji se zasniva na parovima ključeva i vrednosti u string formatu. Primer jednog zapisa predstavljenog JSON formatom:

|  |  |
| --- | --- |
| 1  2  3  4  5 | {"persons":[      {"firstName":"Ben", "lastName":"Hur", "hairColor":"black", "eyeColor":"blue"},      {"firstName":"Ana", "lastName":"Lord", "hairColor":"blue", "eyeColor":"brown"},      {"firstName":"John", "lastName":"David", "hairColor":"brown", "eyeColor":"green"}  ]} |

Objekti se u JSON formatu oivičavaju vitičastim zagradama, a nizovi objekata uglastim zagradama. Ključevi i vrednosti se pišu pod navodnicima, a razdvajaju se karakterom dvotačka (:).

XML je, za razliku od JSON-a, strukturirani format, zato što koristi imenovane elemente. Elementi se sastoje iz otvarajućih i zatvarajućih tagova i vrednosti, a mogu imati i atribute. Tako je XML vrlo sličan HTML-u, a zapravo su i jedan i drugi jezik izdanak istog roditelja. Identičan primer prikazan korišćenjem XML izgleda ovako:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <persons>   <person>    <firstName>Ben</firstName>    <lastName>Hur</lastName>    <hairColor>black</hairColor>    <eyeColor>blue</eyeColor>   </person>     <person>    <firstName>Ana</firstName>    <lastName>Lord</lastName>    <hairColor>blue</hairColor>    <eyeColor>brown</eyeColor>   </person>     <person>    <firstName>John</firstName>    <lastName>David</lastName>    <hairColor>brown</hairColor>    <eyeColor>green</eyeColor>   </person>  </persons> |

**XMLHttpRequest**

JavaScript jezik ima ugrađenu podršku za samostalno upućivanje zahteva i prijem podataka od servera. XMLHttpRequest je JavaScript objekat koji omogućava klijentu razmenu podataka sa serverom i to asinhronim putem.

XMLHttpRequest omogućava ažuriranje podataka, bez potrebe za osvežavanjem kompletne strane. Pre nego što se pozabavimo demonstracijom korišćenja ovog objekta, biće prikazane njegove najznačajnije metode i svojstva. Najznačajnije metode objekta prikazane su u tabeli 29.1

|  |  |
| --- | --- |
| **Metoda** | **Opis** |
| abort() | poništava trenutni zahtev |
| getAllResponseHeaders() | vraća informacije iz zaglavlja |
| getResponseHeader() | vraća tačno određenu informaciju iz zahteva |
| open(method,url,async,uname,pswd) | inicijalizuje zahtev; *method* označava tip HTTP metode koja će biti korišćena; *url* označava adresu na koju se šalje zahtev; *async* je opcioni parametar koji ukazuje na to da li će zahtev biti prosleđen asinhrono ili ne; podrazumevana vrednost je true; *uname* i *pswd* su opcioni parametri koji se mogu koristiti za proces autentifikacije |
| send() | šalje prethodno inicijalizovan zahtev |
| setRequestHeader() | dodaje informacije u zaglavlje zahteva u formi ključeva i vrednosti |

*Tabela 29.1 – Metode objekta XMLHttpRequest*

Nakon pregleda metoda XMLHttpRequest objekta, tabela 29.2 prikazuje njegova najznačajnija svojstva.

|  |  |
| --- | --- |
| **Svojstvo** | **Opis** |
| onreadystatechange | definiše naziv funkcije koja će biti pozvana svaki put kada se vrednost svojstva readyState promeni |
| readyState | sadrži informaciju o statusu XMLHttpRequest objekta; stanja se izražavaju brojevima od 0 do 4: 0: zahtev nije inicijalizovan 1: konekcija sa serverom ostvarena 2: zahtev primljen 3: zahtev u procesu obrade 4: obrada zahteva završena i odgovor spreman |
| responseText | sadrži podatke odgovora u formi stringa |
| responseXML | sadrži podatke odgovora u formi XML-a |
| status | sadrži statusnu informaciju izraženu HTML statusnim kodom |
| statusText | sadrži tekst statusa |

*Tabela 29.2 – Svojstva objekta XMLHttpRequest*

Nakon upoznavanja sa osobinama XMLHttpRequest objekta, sledi jednostavan primer njegove upotrebe.

HTML stranice biće sledeći:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <html>  <head>   <title>JavaScript Demo</title>  </head>  <body>   <h2>JavaScript Demo</h2>   <button type="button" id="getBtn">Get Content</button>   <div id="contentContainer"></div>    </body>  </html> |

HTML kod se sastoji iz button elementa za pokretanje asinhronog učitavanja i div elementa za prikaz sadržaja koji će biti asinhrono učitan.

Kao sadržaj koji će biti asinhrono učitan, poslužiće sadržaj jednog običnog tekstualnog fajla sa nazivom content.txt, koji je potrebno da se nađe u istom folderu u kome se nalazi i .html fajl.

Kako bi se objekat XMLHttpRequestuposlio, poslužiće sledeći kod:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <script type="text/javascript">    window.onload = function () {   var button = document.getElementById("getBtn");     button.addEventListener("click", function () {      var xhttp = new XMLHttpRequest();      xhttp.onreadystatechange = function () {     if (xhttp.readyState == 4 && xhttp.status == 200) {     document.getElementById("contentContainer").innerHTML = xhttp.responseText;     }    };    xhttp.open("GET", "content.txt", true);    xhttp.send();   });  }  </script> |

Prikazanim JavaScript kodom definisana je logika za registrovanje event handlera za klik na buttonelement sa id vrednošću *getBtn*. Klikom na ovaj element, pokreće se inicijalizacija asinhronog zahteva, kreiranjem objekta XMLHttpRequest.

Nakon kreiranja objekta XMLHttpRequest, registruje se event handler za promenu vrednosti stanja ovog objekta. Malopre je pomenuto da ovaj objekat može biti u jednom od 5 stanja (pogledajte tabelu 29.2). Kada je objekat u poslednjem stanju, a status zahteva sa kodom 200, znači da je obrada zahteva uspešno završena i da su podaci koji su asinhrono učitani spremni za prikaz ili dalju obradu. U primeru se podaci samo prikazuju unutar div elementa sa id-jem contentContainer.

Zahtev je, naravno, potrebno i inicijalizovati i poslati, i to se postiže linijama u kojima se pozivaju metode open i send.

U primeru se podaci učitavaju iz tekstualnog fajla, kako bi primer bio što lakši. U realnim situacijama, umesto naziva tekstualnog fajla, uglavnom se navodi adresa do fajla na serveru koji sadrži serversku skriptu, koja izvršava određenu logiku i isporučuje podatke.

**Slanje podataka korišćenjem XMLHttpRequest objekta**

Prethodni primer ilustruje način na koji se obavlja prijem podataka korišćenjem XMLHttpRequestobjekta. Pored prijema podataka, XMLHttpRequest omogućava i njihovo slanje.

Kako bi se podaci slali korišćenjem XMLHttpRequest objekta, dovoljno je send metodi proslediti podatke koje je potrebno da pošalje na određeni URL, definisan kao parametar open metode. Primer ilustruje blok koda koji obavlja slanje jednog Person objekta u XML formatu na server:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var xmlData = "<Person firstname='Rick' lastName='Delorme' hairColor='Brown'  eyeColor='Brown' /> ";  var xReq = new XMLHttpRequest();  xReq.open("POST", "/ReceiveXMLData.aspx", false);  xReq.responseType = "text";  xReq.send(xmlData); |

S obzirom da se ovoga puta podaci emituju na server, kao metoda slanja zahteva se koristi HTTP metoda POST.

**Serijalizacija i deserijalizacija JSON podataka**

Pojmovi serijalizacije i deserijalizacije podataka odnose se na proces konvertovanja objekata nekog jezika u i iz formata koji se koristi za njihovo transportovanje kroz mrežu. U našoj situaciji, podaci se nalaze u obliku JavaScript objekata, a potrebno ih je konvertovati u JSON i obrnuto.

Većina browsera omogućava izvornu podršku za rad sa JSON i XML podacima. Tako se za serijalizaciju i deserijalizaciju JSON podataka može koristiti JSON objekat.

U jednom od prethodnih primera u ovoj lekciji, korišćenjem XMLHttpRequest objekta pročitani su podaci iz jednog txt fajla. Podaci nisu bili ni u jednom specijalnom formatu. Hajde da sada vidimo kako obaviti identičan posao, ali ovoga puta sa podacima u JSON formatu. U narednom primeru će biti pročitani podaci iz fajla sa ekstenzijom .json, koji ima sledeći sadržaj:

|  |  |
| --- | --- |
| 1 | {"firstName":"Ben", "lastName":"Hur", "hairColor":"black", "eyeColor":"blue"} |

Kod:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | <html>  <head>   <title>JavaScript Demo</title>  </head>  <body>   <h2>JavaScript Demo</h2>   <button type="button" id="getBtn">Get Content</button>   <div id="contentContainer"></div>     <script type="text/javascript">      window.onload = function () {     var button = document.getElementById("getBtn");       button.addEventListener("click", function () {        var xhttp = new XMLHttpRequest();        xhttp.onreadystatechange = function () {       if (xhttp.readyState == 4 && xhttp.status == 200) {          var jsonPerson = JSON.parse(xhttp.responseText);          console.log("Firstname: " + jsonPerson.firstName);        console.log("Lastname: " + jsonPerson.lastName);        console.log("Hair color: " + jsonPerson.hairColor);        console.log("Eye color: " + jsonPerson.eyeColor);       }      };      xhttp.open("GET", "content.json", true);      xhttp.send();       });      }     </script>    </body>  </html> |

Ovoga puta se, kada se podaci uspešno prime, vrši deserijalizacija dobijenih podataka, tako što se oni iz JSON formata prevode u objekte jezika JavaScript. Za obavljanje takvog posla koristi se metoda parse objekta JSON. Ovoj metodi se prosleđuju podaci u JSON formatu.

Metoda parse u potpunosti obavlja deserijalizaciju podataka, tako da su oni, nakon završetka logike metode parse, dostupni kroz JavaScript objekat koji je referenciran promenljivom jsonPerson.

Deserijalizacijom, ključevi postaju nazivi svojstava, a vrednosti ključeva vrednosti kreiranih svojstava. Tako je podacima moguće vrlo lako pristupiti čitanjem svojstava kreiranog objekta.

U primeru se, korišćenjem opisane tehnike, vrši čitanje vrednosti svojstava parsiranog objekta, tako da prikazani kod proizvodi sledeći rezultat:

|  |  |
| --- | --- |
| 1  2  3  4 | Firstname: Ben  Lastname: Hur  Hair color: black  Eye color: blue |

Proces serijalizacije obrnut je proces procesu deserijalizacije. Procesom serijalizacije, podaci se prevode iz objektnog modela u format koji je pogodan za čuvanje ili prenos preko mreže. Sledeći primer ilustruje serijalizaciju jednog JavScript objekta u JSON format:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | var person = {   firstName: "Ben",   lastName: "Hur", hairColor: "black",   eyeColor: "blue"    };    var jsonPerson = JSON.stringify(person);  console.log("Person in JSON format: " + jsonPerson); |

U prikazanom primeru, prvo se kreira jedan JavaScript objekat sa četiri svojstva. Da bi se izvršila serijalizacija, koristi se objekat JSON i njegova metoda stringify. Ovoj metodi se prosleđuje objekat koji je potrebno serijalizovati, odnosno, u ovom slučaju prevesti u JSON format.

Metoda stringify kao povratnu vrednost emituje string koji predstavlja prosleđeni objekat konvertovan u JSON format. U primeru se serijalizovani objekat smešta unutar promenljive sa nazivom jsonPerson, i vrednost takve string promenljive se emituje na izlazu. Tako primer proizvodi sledeći izlaz:

|  |  |  |
| --- | --- | --- |
| 1 | | Person in JSON format: {"firstName":"Ben","lastName":"Hur","hairColor":"black","eyeColor":"blue"} |
| **Napomena**    Pojedini primeri  iz ove nastavne jedinice možda neće raditi u nekim (posebno starijim) web browserima. |