JavaScript

Deseti dio

Pregled

- Ajax
  - Uvod
  - Kako radi
  - XMLHTTPRequest objekat
  - Slanje zahtjeva serveru
  - Prihvatanje odgovora sa servera
  - XML uvod
  - Komunikacija sa backend sktiptama
- JSON
  - Uvod
  - Sintaksa
  - JSON vs XML
  - Parse
  - Stringify

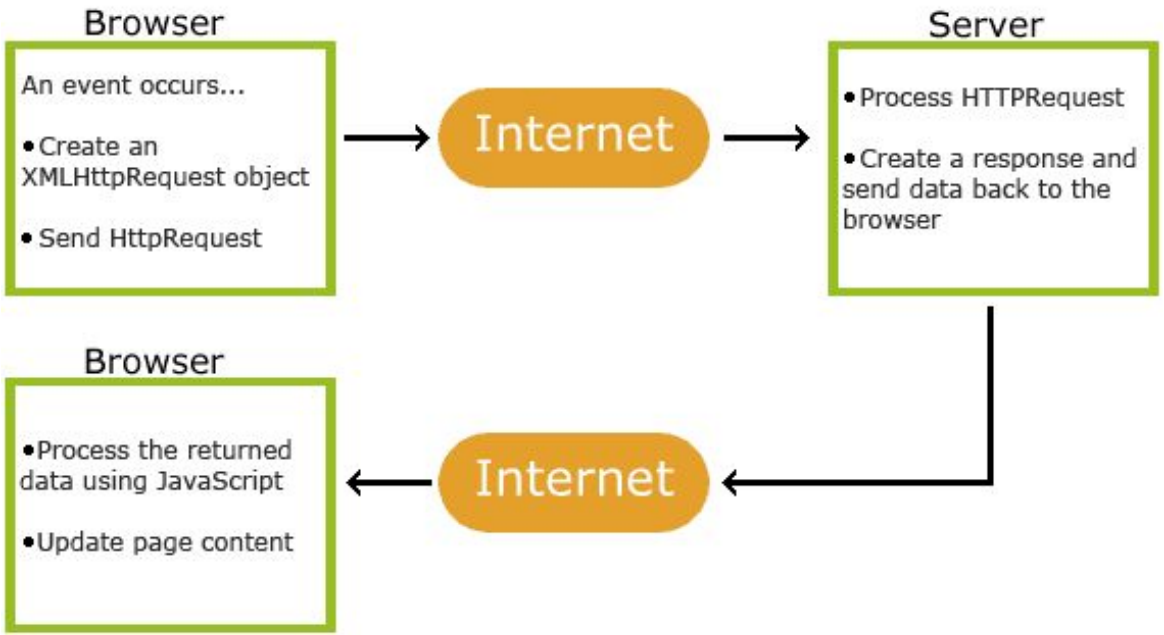
Obnavljanje i napomene

- Test 3 - subota u 11h
- Domaći 4.
- Domaći 5.
- Šta smo rekli, čemu služi Closure, kad se koristi, kako se kreira
- Šta predstavlja module pattern i koja je njegova glavna funkcija
- ByRef, ByValue
- Week 9, code - 1, da nacrtamo kako se izvršava kod

Ajax, uvod

- Odakle smo do sad uzimali podatke, a šta smo rekli,od koga se podaci najčešće traže?
- AJAX(Asynchronous JavaScript and XML) nastao iz sledećih razloga:
  - Omogućava čitanje podataka sa web servera - nakon što se stranica učita
  - Ažurira web stranicu bez reloading
  - Slanje podataka serveru (u pozadini)
- Ajax nije programski jezik, a koristi kombinaciju:
  - XMLHTTPRequest objekta (slanje zahtjeva web serveru)
    - Ovaj objekat je dio browser-a (built-in), tj. dio Web Browser-a APIa
- Naziv ajax nas malo pogrešno navodi da samo XML dokumenti mogu biti prenošeni putem njega. To nije tačno jer putem AJAX-a moguće je razmjenjivati tekstualne fajl, JSON, itd.
- AJAX omogućava web stranicama da budu ažurirane asinhrono razmjenom podataka sa web serverom (u pozadini, kao Timer, pomoću Web APIa). Ovo znači da je moguće ažuriratu web stranicu, a da pri tom nije potrebno da se cijela stranica refresh-uje

Ajax, kako radi



Ajax, XMLHTTPRequest objekat

- Ujedno i srž AJAXa đ
- Svi moderni pretraživači podržavaju ovaj objekat
- Služi za razmjenu podataka sa web serverom.
- SIntaksa za kreiranje XMLHTTPRequst objekta

```
var xhttp = new XMLHttpRequest();
```

- Zbog bezbjenosti, moderni pretraživači onemogućuju pristup između različitih domena
  - To znači da i web stranica, kao i fajl koji pokušamo da učitamo, moraju biti sačuvani na istom serveru
- Na starim pretraživačimaovaj objekat se kreira malo drugačije

```
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
```

Ajax, XHR metodi

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method, url, async, user, psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

Ajax, XHR svojstva (properties)

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Ajax, slanje zahtjeva serveru

- Pomoću Ajax-a možete da šaljete zahtjeve serveru.
- To je najčešće radi u dva koraka

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

- Metod open() služi da definišemo request
  - Prvi argument predstavlja tip zahtjeva (**GET** ili **POST**)
    - Sjećate li se kada koristimo GET, a kada POST
  - Drugi argument predstavlja lokaciju fajla
  - Treći argument služi da definišemo da li je izvršavanje ovog metoda asinhrono (true ako jeste, i preporuka je da uvijek bude true)
- Metod send() služi za slanje zahtjeva serveru
  - Ako imamo POST request metod send najčešće ima argument(string), koji predstavlja podatke koje šaljemo serveru (npr. da upiše nešto u bazu)

Ajax, slanje zahtjeva GET ili POST

- GET je jednostavniji i brži nego POST, međutim uvijek koristite POST request kada:
  - Cached file nije opcija (kada ažuriramo fajl ili bazu na serveru)
  - Kada šaljemo veće količine podataka na server (POST nema size limit)
  - Kada šaljemo korisničke unose (user input) - mogu da sadrže nepoznate karaktere
    - POST je bezbjedniji nego GET
- **Pogledajmo prvi primjer (code-1)**
- Takođe moguće je poslati web serveru i sledeće
  - Primjer 1
  - Primjer 2
- Sjećate se dijela header dok HTTP ?
  - Više o headerima na linku
    - U backend kursu, a dijelom i u frontend, radićemo puno detaljnije Header
  - Pogledajmo Primjer 3
  - Koristi se XHR metod **setRequestHeader(*header*, *value*)**

Ajax, url i async parametri za open()

- URL parametar kod metoda open() je lokacija fajla na serveru
  - Može da bude bili koji tip fajla, npr. txt, xml, json fajlovi ili fajlovi koji se izvršavaju na serveru kao što je .php ili .py, itd. Fajlovi koji se izvrašvaju na server (backend) izvršavaju akcije na serveru prije nego što se odgovor pošalje klijentu (frontend)
- Zahtjevi koji se šalju serveru gotovo uvijek bi trebalo da se šalju asinhrono
  - Asinhronim slanjem, JS ne mora da čeka odgovor od servera, tj. može da nastavi da izvšava druge skripte, pa tek kada stigne odgovor, da odradi neku akciju nad njim
  - Kako znamo da je odgovor stigao sa servera?
    - Na xhr zakačimo event "**readystatechange**"
    - O ovome ćemo uskoro da govorimo detaljnije
  - Ako se zahtjev šalje sinhrono (danas se najčešće koristi samo za testiranje), postavimo vrijednost trećeg parametra open() metoda na **false** čime zaustavljamo izvršavanje JS koda sve dok se open() metod ne izvrši do kraja
    - Ovo je jako loše koristi u praksi, pretposavljam da možete odma navesti bar 3 razloga zašto je ovo loše
    - Pogledajmo primjer

Ajax, prihvatanje odgovora sa servera

- Klučna svojstva (properties) XHR objekta za prihvatanje odgovora sa servera

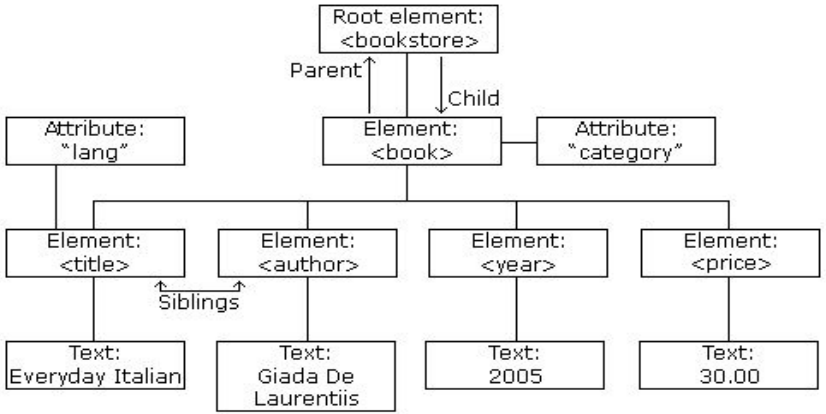
Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Ajax, prihvatanje odgovora sa server, nastavak

- Za potpuniju listu status poruka XHR objekta možete pročitati sledeći link
- Šta mislite, koliko puta će se pozvati event handler funkcija event-a “readystatechange” XHR objekta?
- Ako imamo više od jednog AJAX taska na stranici, trebalo bi krerati jednu funkciju za izvršavanje XHR, a za svaki AJAX task jednu callback funkciju
- Neki od response metoda i property-a xhr objekta
    - responseText - vraće response podatke kad JS string
    - responseXML - vraće response podatke kao XML
        - XML DOM moguće koristiti
    - getResponseHeader(header) - vraće informacije o određenom header-u sa servera sa kog je stigao odgovor
    - getAllResponseHeaders() - vraće informacije o svim header-ima sa servera sa kog je stigao odgovor
- **Pogledajmo primjer (code-2)**

Ajax, XML - uvod

- Vrlo slična struktura kao i HTML fajl, slična terminologija, svi elementi su custom XML elementi
- Osnovna primjena je čuvanje podataka u fajlu koji je razumljiv i ljudima i računaru.
- Nastavak



Ajax, komunikacija sa backend skriptama

- Osim što Ajax koristimo za traženje txt, xml, json fajlova, omogućava nam da komuniciramo i sa skriptama koje se izvrašavaju na serverima, tj. Kažemo da komuniciramo sa backend-om
- Čisto pogledajmo par primjera da dobijete osnovu sliku o tome gdje je prisutpna velika primjena Ajax-a
  - Primjer 1
  - Ako vas zanima i PHP dio
    - Link
- Sve ovo ćemo raditi mnogo detaljnije kada budemo prešli na naprednije kurseve, ReactJS (frontend) i NodeJS (backend)
- **Pogledajmo još jedan primjer (code-3)**

JSON, uvod

- Gotovo identićna uloga kao i XML
  - Glavna uloga čuvanje podataka na razumljiv način
- JSON je tekst zapisan kao JS objekat
- Kada se podaci razmjenjuju između pretraživača i servera, ti podaci moraju da budu tekstualni
  - JSON je tekst, a svaki JS objekat možemo da konvertujemo u JSON i pošaljemo JSON serveru
  - Takođe, bilo koji JSON možemo da konvertujemo u JS objekat
  - Na ovaj način možemo da radimo sa podacima kao JS objektima, bez komplikovanog parsiranja i prevoda
- Sa JSON podacima moguće je:
  - Slanje
  - Primanje
  - Čuvanje
- JSON (JavaScript Object Notation) jednostavan, čitljiv, format za razmjenu podataka
  - Originalno definisan od osnivača JSa (Douglas Crockford)
- Napoma: JSON je tekst, što znači da je nezavisan od programskih jezika
  - Koristi JS sintaksu, ali JSON format je samo tekst

JSON, uvod (1)

- Zašto JSON
  - Kako je JSON format tekst, vrlo se jednostavno šalje do i sa servera
  - Koristi se kao format za čuvanje podataka kod bilo kog programskog jezika
- JS ima ugrađene funkcije koje konvertuju string, napisan u JSON formatu, u JS objekat
  - JSON.parse()
- Kada stignu podaci sa serveru, u JSON formatu, ti podaci mogu da se koriste kao bilo koji JS objekat
- JSON sintaksa je podskup JS sintakse

JSON, sintaksa

- Izvedena iz notacije objekata kod JSa
  - Podaci se čuvaju u paru key:value
  - Podaci se odvajaju zarezom
  - {} čuvaju objekat
  - [] čuvaju niz
- Key (name) se postavlja u “”
  - Kod JSa to nije obavezno, kod JSONa jeste !
  - Može li jednostruki navodnik
  - Pogledajmo
- Kod JSON-a, vrijednosti moraju biti jedan od sledećih tipova podataka
  - **String(moraju da koriste double quotes), broj, objekat (JSON objekat), niz, boolean, null**
  - Tipovi kao što su function, date i undefined nisu podržani u JSONu

JSON vs XML

- Kako bi ovo izgledalo u XML formatu

JSON Example

```
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

JSON vs XML, nastavak

- JSON je kao XML jer
  - JSON i XML čitljivi sa ljude (self descibing)
  - JSON i XML imaju hijerarhiju (values within values)
  - JSON i XML se mogu parsirati i koristiti za više programskih jezika
  - JSON i XML mogu biti preuzeti sa XHR
- Po čemu se JSON razlikuje od XMLa
  - JSON ne korisiti end tag
  - JSON je kraći
  - JSON se brže čita i upisuje
  - JSON može da sadrži niz
- Najveća razlika XML mora da bude parsiran pomoću XML parsera dok JSON može biti parsiran standardom (ugrađenom) JS funkcijom. Ova razlika nam jasno govori da je JSON bolji nego XML
- Kod AJAX aplikacija, JSON je brži i jednostavniji od XMLa
  - Kod XMLa: pokupiti XML dokument, primijeniti XML DOM da se prođe kroz XML dokument, izvuci vrijednosti i sačuvaj ih u varijablu
  - Kod JSONa: pokupiti JSON string, JSON.parse za parsiranje JSON stringa

JSON Parse

- Kada podaci stignu sa server, uvijek stignu u string formatu
- Da bi ste jednostavno pretvorili JSON string u JS objekat koristi se ugrađena JS funkcija
  - JSON.parse()
- Isprobajmo primjer
- **Pogledajmo sada code-4**
  - Kako da štampamo ime trećeg ljubimca?
- Pogledajmo i primjer sa datumom
  - Primjer
- Pogledajmo primjer sa funkcijom
  - Primjer
  - Ovo vrlo opasno koristiti, moguće ali nije dobra praksa
  - Eval metod izvršava JS kod koji je zapisan u vidu stringa

JSON stringify

- Kada šaljemo podatke ka serveru, ti podaci moraju biti zapisani u vidu stringa
  - JSON.stringify() ugrađeni JS metod koji konvertuje JS objekat u string
- Pogledajmo primjer
  - Primjer
  - Šta ako funkciju ne konvertujemo u string
  - Šta ako umjesto funkcije za vrijednost **age** property-a stavimo undefined vrijednost
- Pogledajmo još jedan primjer
  - Primjer
- Za još detalja vezanih za JSON pročitati
  - JSON Objects
  - JSON Arrays
  - JSON PHP
  - JSON HTML
  - JSONP

Pitanja