AJAX

AJAX je skraćenica za "*Asynchronous JavaScript and XML*" (Asinhroni JavaScript i XML). Ova tehnologija omogućuje slanje i primanje podataka između web pregledača i servera asinhrono, što znači da se podaci mogu razmenjivati sa serverom u pozadini bez potrebe za osvežavanjem cele stranice.

Jedan od najčešćih primera primene AJAX-a je dinamičko učitavanje sadržaja, kao što su podaci iz baze podataka ili rezultati pretrage, bez osvežavanja cele stranice. Na primer, možemo koristiti AJAX za učitavanje novih postova na društvenoj mreži dok korisnik lista svoju početnu stranicu, bez potrebe za osvežavanjem cele stranice svaki put kada korisnik skroluje.

AJAX je moćna tehnologija koja omogućava interaktivne i dinamičke web stranice. Razumevanje osnova rada sa AJAX-om omogućava programerima da razvijaju moderna i efikasna web rešenja koja poboljšavaju korisničko iskustvo. Uz pravilno korišćenje, AJAX može značajno poboljšati performanse i funkcionalnosti web aplikacija.

AJAX UZ JQUERY

U prvom polugodištu smo započeli temu o jQuery biblioteci (ako se neko seća 😊) koju ćemo takođe koristiti i u drugom polugodištu.

jQuery je popularna JavaScript biblioteka koja olakšava manipulaciju DOM-om, upravljanje događajima, animacije i mnoge druge zadatke. Takođe, jQuery pruža jednostavnu i efikasnu podršku za rad sa AJAX-om. U ovoj lekciji ćemo istražiti kako koristiti jQuery za izvršavanje AJAX zahteva.

Pre nego što počnemo koristiti jQuery za AJAX, prvo je potrebno uključiti jQuery biblioteku u HTML dokumentu. To se može uraditi uključivanjem jQuery skripte iz eksternog izvora ili preuzimanjem jQuery datoteke i uključivanjem lokalno.

Izvršavanje AJAX zahteva pomoću jQuery

Kada je jQuery uključen, možemo koristiti `$.ajax()` funkciju za slanje AJAX zahteva. Ova funkcija omogućava različite parametre za konfigurisanje zahteva, uključujući URL endpoint, tip zahteva (GET, POST, itd.), podatke koji se šalju na server i funkciju za obradu odgovora.

```
$.ajax({
  url: 'https://api.example.com/data',
  method: 'GET',
  success: function(response) {
      // Obrada uspešnog odgovora
      $('#rezultat').html(response);
  },
  error: function(xhr, status, error) {
      // Obrada greške
      console.error('Došlo je do greške prilikom slanja zahteva:', status, error);
  }
});
```

Napomena: URL http://www.example.com/data se koristi kao primer, ali to nije stvarna adresa na internetu. Stoga, umesto stvarnih podataka, možete koristiti URL koji vodi ka stvarnim podacima sa kojima želite da radite u svojoj aplikaciji. Na primer, možete koristiti URL ka nekom API-ju koji pruža podatke u JSON formatu.

Metode GET i POST

Metode GET i POST su dve od osnovnih HTTP metoda koje se koriste za slanje zahteva sa klijenta (npr. web pregledača) na server. Obe metode imaju svoje specifične svrhe i načine korišćenja.

Metoda GET

Metoda GET se koristi za zahteve koji traže od servera određene resurse ili informacije. Ovi zahtevi se šalju kao deo URL adrese i obično se koriste za dohvativanje podataka. Kada koristite GET metodu, podaci se šalju kao deo URL adrese u tzv. query string-u, što znači da su vidljivi u URL-u.

Na primer, kada koristite pretraživač i unesete URL adesu u adresnu traku ili kliknete na link, to je obično GET zahtev. Ova metoda se takođe često koristi za čitanje podataka sa servera, poput preuzimanja slika, članaka ili drugih resursa.

Metoda POST

Metoda POST se koristi za slanje podataka na server kako bi se izvršila određena akcija, kao što je unos podataka u bazu podataka ili slanje forme. Za razliku od GET metode, podaci se ne šalju kao deo URL adrese, već se šalju u HTTP telu zahteva.

Ova metoda se često koristi kada želite da pošaljete veće količine podataka ili kada želite da podaci budu skriveni od javnosti, budući da se ne pojavljuju direktno u URL-u.

Upotreba

`GET`: Koristi se za dohvativanje podataka, čitanje informacija i kada ne želite da podaci budu skriveni. Na primer, pretraživanje po internetu ili prikazivanje sadržaja stranice.

`POST`: Koristi se za slanje podataka na server, kao što su forme ili podaci za kreiranje ili ažuriranje resursa. Na primer, slanje podataka iz forme za registraciju ili slanje komentara na blogu.

`success`: Ovo je naziv ključa u objektu koji se koristi za definisanje funkcije koja će se izvršiti kada se uspešno primi odgovor od servera. `success` ključ predstavlja deo konfiguracije AJAX poziva koji označava da se želi definisati funkcija koja će se izvršiti kada se zahtev uspešno izvrši.

`function(response)`: Ovo je anonimna funkcija koja će biti izvršena kada se uspešno primi odgovor od servera. Ova funkcija obično ima jedan argument, obično nazvan response, koji

predstavlja podatke koje server vraća kao odgovor na AJAX zahtev. U ovom argumentu ćete obično dobiti tekst od servera (npr. HTML, XML, JSON, itd.), u zavisnosti od toga kako je server konfigurisan da vraća odgovor.

U primeru na početku lekcije, kada se uspešno primi odgovor od servera, izvršava se anonimna funkcija definisana unutar `success` ključa. Ova funkcija postavlja HTML sadržaj elementa sa ID-jem `#rezultat` na vrednost odgovora koji je primljen od servera. Ovo omogućava ažuriranje dela stranice sa novim podacima koje je server vratio.

`error`: Ovo je naziv ključa u objektu koji se koristi za definisanje funkcije koja će se izvršiti ako dođe do greške prilikom slanja zahteva ka serveru ili ako server vrati neuspešan odgovor. error ključ predstavlja deo konfiguracije AJAX poziva koji označava da se želi definisati funkcija koja će se izvršiti ako dođe do greške.

`xhr`: Ovaj argument predstavlja XMLHttpRequest objekat koji se koristi za slanje zahteva ka serveru. Možete koristiti ovaj objekat da pristupite dodatnim informacijama o grešci ili zahtevu.

`status`: Ovaj argument predstavlja HTTP status kod koji je server vratio kao odgovor na zahtev. Na primer, 404 označava da resurs nije pronađen, 500 označava internu serversku grešku, itd.

`error`: Ovaj argument predstavlja tekstualni opis greške koji se vraća ako dođe do greške prilikom slanja zahteva ili ako server vrati neuspešan odgovor.

U našem primeru, kada dođe do greške prilikom slanja zahteva ili ako server vrati neuspešan odgovor, izvršava se anonimna funkcija definisana unutar `error` ključa. Ova funkcija ispisuje poruku o grešci zajedno sa HTTP status kodom i tekstualnim opisom greške u konzoli pregledača. Ovo omogućava programeru da vidi šta je uzrok greške i da preduzme odgovarajuće akcije za rešavanje problema.