АКАДЕМИЈА ТЕХНИЧКО-ВАСПИТАЧКИХ СТРУКОВНИХ СТУДИЈА ОДСЕК НИШ

SERVERSKE TEHNOLOGIJE

Vežba 2 – AJAX (Asynchronous JavaScript and XML)

23.10.2024.

Teorijske napomene

AJAX (Asynchronous JavaScript and XML) predstavlja skup tehnika koje pomažu da se aplikacija poveže sa serverom, pošalje zahtev i dobije odgovor bez ponovnog učitavanja cele stranice. Ovo omogućava korisnicima da brže interaguju sa aplikacijom, jer se samo potrebni podaci ažuriraju, bez osvežavanja čitave stranice. Uprkos nazivu, AJAX se ne oslanja isključivo na XML; danas se češće koristi **JSON** format zbog lakše čitljivosti i manje veličine podataka.

AJAX se oslanja na JavaScript i koristi **XMLHttpRequest** objekat za slanje i primanje podataka sa servera. AJAX proces se obično može podeliti u sledeće korake:

1. **Kreiranje AJAX zahteva**: Na klijentskoj strani, **JavaScript** se koristi za kreiranje **HTTP** zahteva koji sadrži podatke o tome šta želimo od servera (npr. provera korisničkog imena, pretraga podataka, ažuriranje informacija).
2. **Slanje zahteva**: Zahtev se šalje serveru bez osvežavanja stranice. Koriste se metode kao što su **GET** (za čitanje podataka) ili **POST** (za slanje podataka, obično u formama).
3. **Obrada zahteva na serveru**: Server prima zahtev, obrađuje ga (npr. proverava bazu podataka, obavlja neku funkciju), i vraća odgovor klijentu.
4. **Obrada odgovora**: JavaScript na klijent strani prima odgovor i koristi ga za ažuriranje delova stranice bez ponovnog učitavanja.

Pre AJAX-a, za svaki zahtev prema serveru (npr. prijava na sistem) bila bi potrebna nova stranica ili ponovno učitavanje, što je usporavalo korisničko iskustvo. AJAX omogućava:

- **Ažuriranje stranice u realnom vremenu**
- **Smanjenje opterećenja servera**
- **Bolje korisničko iskustvo**

Kada koristiti AJAX?
AJAX se koristi u situacijama gde je potrebno ažurirati određeni deo stranice ili dobiti informacije iz servera bez potrebe za osvežavanjem. Najčešće primene su:

- Autocomplete polja za pretragu (popunjavanje predloga dok korisnik kuca)
- Dinamičko učitavanje sadržaja (kao što su komentari ili obaveštenja)
- Ažuriranje korpe u e-commerce aplikacijama
- Validacija podataka, kao što je provera dostupnosti korisničkog imena, ili provera da li je email adresa već registrovana

Praktični primer 1: Pretraga proizvoda u realnom vremenu

U ovom primeru simuliraćemo pretragu proizvoda u realnom vremenu, gde AJAX dinamički šalje unos serveru i prikazuje odgovarajuće rezultate bez potrebe za osvežavanjem stranice. Ovaj pristup omogućava korisnicima trenutne povratne informacije, što značajno unapređuje interaktivnost i celokupno korisničko iskustvo. Da bismo pojednostavili implementaciju, koristićemo statički JSON fajl sa osnovnim podacima o proizvodima, omogućavajući da se celokupan proces pretrage odvija direktno u JavaScript-u, bez dodatne serverske logike.

1. HTML i CSS

Kreiramo osnovnu strukturu koja sadrži polje za pretragu i sekciju za prikaz rezultata. Stilizovanje je preporučljivo da bude u posebnom fajlu (style.css), ali za potrebe ovog primera možete i direktno na HTML stranici ubaciti par naredbi.

```
<!DOCTYPE html>
<html lang="sr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pretraga proizvoda</title>
    <style>
        /* Stil za rezultate */
        .result-item {
            padding: 10px;
            border-bottom: 1px solid #ccc;
        }
        .result-item:last-child {
            border-bottom: none;
        }
    </style>
</head>
<body>
    <h2>Pretraži proizvode</h2>
    <input type="text" id="search" placeholder="Unesite ime proizvoda"
                                                onkeyup="searchProducts()">
    <div id="results"></div>
    <script src="script.js"></script>
</body>
</html>
```

Polje za unos (**id="search"**) reaguje na događaj **onkeyup**, pozivajući funkciju **searchProducts()** svaki put kada korisnik unese karakter, a div **results** prikazuje rezultate pretrage.

2. JSON podaci o proizvodima

Pravimo JSON fajl (products.json) sa podacima o proizvodima. U pravoj aplikaciji, ovaj fajl bi bio na serveru, ali ćemo ga za ovaj primer učitati lokalno.

```
[
    {"name": "Laptop"},
    {"name": "Telefon"},
    {"name": "Monitor"},
    {"name": "Tastatura"},
    {"name": "Miš"},
    {"name": "Punjač"}
]
```

Ovaj fajl sadrži niz proizvoda sa samo jednom informacijom o svakom - imenom proizvoda. Naravno, možemo ga proširiti i cenama proizvoda, proizvođačima, godinama proizvodnje, itd. Proširena verzija fajla bi na primer ovako izgledala:

```
[
    {"name": "Laptop", "price": "800 EUR"},
    {"name": "Telefon", "price": "400 EUR"},
    {"name": "Monitor", "price": "200 EUR"},
    {"name": "Tastatura", "price": "50 EUR"},
    {"name": "Miš", "price": "20 EUR"},
    {"name": "Punjač", "price": "15 EUR"}
]
```

3. JavaScript za pretragu i prikaz rezultata

U script.js ćemo implementirati funkciju searchProducts() koja učitava products.json, filtrira rezultate na osnovu unetog teksta i prikazuje ih u realnom vremenu.

```
// Funkcija za pretragu proizvoda
function searchProducts() {
    const query = document.getElementById("search").value.toLowerCase();
    const resultsContainer = document.getElementById("results");

    // Čistimo rezultate pretrage pre svake nove pretrage
    resultsContainer.innerHTML = '';

    // AJAX zahtev za učitavanje JSON podataka
    const xhr = new XMLHttpRequest();
    xhr.open("GET", "products.json", true);
```

```
    xhr.onload = function() {
        if (xhr.status === 200) {
            const products = JSON.parse(xhr.responseText);
            const filteredProducts = products.filter(product =>
                product.name.toLowerCase().includes(query)
            );

            // Prikaz filtriranih rezultata
            if (filteredProducts.length > 0) {
                filteredProducts.forEach(product => {
                    const item = document.createElement("div");
                    item.classList.add("result-item");
                    item.textContent = product.name;
                    resultsContainer.appendChild(item);
                });
            } else {
                resultsContainer.innerHTML = '<p>Nema rezultata</p>';
            }
        }
    };
    xhr.send();
}
```

searchProducts funkcija:

- Preuzima unos korisnika iz polja i konvertuje ga u mala slova za lakše poređenje.
- Čisti rezultate prethodne pretrage (resultsContainer.innerHTML = '') pre nego što prikaže nove rezultate.

AJAX zahtev:

- Kreira AJAX zahtev koji otvara products.json koristeći **GET** metodu.
- Kada je fajl učitan, podaci se parsiraju iz JSON-a i skladište u promenljivu **products**.

Filtriranje proizvoda:

- Metoda **.filter()** koristi unos korisnika kao kriterijum za filtriranje proizvoda. Proizvodi koji sadrže unos korisnika (korišćenjem **.includes()**) biće prikazani.

Prikaz rezultata:

- Ako takvi filtrirani proizvodi postoje, za svaki se kreira ** ** sa klasom **result-item**, a ime proizvoda se prikazuje u **resultsContainer**.
- Ako nema rezultata, prikazuje se poruka **"Nema rezultata"**.

Kako sve radi zajedno?

Kada korisnik unese tekst u polje za pretragu:

1. Funkcija searchProducts() poziva AJAX zahtev za preuzimanje products.json.
2. Prikazuje se proizvodi čije ime sadrži uneti tekst, bez potrebe za osvežavanjem stranice.
3. Rezultati se ažuriraju dok korisnik kuca, pružajući mu interaktivno iskustvo pretrage.

Zadatak za samostalni rad na ovom primeru:

1. Proširenje JSON fajla

- U JSON fajlu **products.json**, dodajte malo više informacija o proizvodima. Osim naziva proizvoda (name), uključite i dodatne podatke, kao što su:
    - Cena proizvoda (price)
    - Proizvođač (manufacturer)
    - Kategorija proizvoda (category)

2. Izmena funkcionalnosti pretrage

- Prilagodite funkciju searchProducts() tako da omogućite pretragu po više parametara. Umesto samo imena proizvoda, omogućite pretragu po:
    - Nazivu proizvoda
    - Ceni
    - Proizvođaču
- Filtrirajte rezultate na osnovu unetih ključnih reči za ove parametre.

3. Stilizovanje rezultata

- Prebacite sav stil u poseban CSS fajl, a onda dodajte stilove koji će omogućiti da rezultati izgledaju pregledno i jasno. Koristite stilove kao što su boje pozadine, margine, tabele...
- Takođe, dodajte stil i za **nema rezultata** poruku, da bi bila jasno prikazana korisnicima.

BONUS:

Debounce tehnika se koristi da se smanji broj poziva funkcija, posebno kada korisnik brzo unosi podatke. Kada koristimo AJAX u pretrazi, pozivanje servera na svaki unos može izazvati previše zahteva, što može usporiti aplikaciju. Debounce tehnika omogućava da funkcija bude pozvana samo nakon što korisnik prestane da kuca određeno vreme (npr. 500 milisekundi), što smanjuje broj nepotrebnih zahteva. Ovu vežbu možete nadograditi tom tehnikom i dobiti bonus poene.

U nastavku je dat primer **debouncedSearch** funkcije koja koristi **setTimeout** da zakaže pozivanje funkcije **searchProducts()** nakon što korisnik prestane da kuca 500 milisekundi. Ukoliko korisnik unese novi karakter pre nego što se prošli zahtev završi, prethodni zahtev je otkazan pomoću **clearTimeout.**

```
// Debounce funkcija koja čeka 500ms pre pozivanja searchProducts
        function debouncedSearch() {
            clearTimeout(debounceTimeout); // Brisanje prethodnog tajmera
            debounceTimeout = setTimeout(() => {
                searchProducts(); // Pozivanje funkcije za pretragu
            }, 500); // Pozivanje nakon 500ms
        }
```

Praktični primer 2: Provera dostupnosti korisničkog imena

U ovom primeru ćemo implementirati formu koja omogućava korisnicima da provere da li je korisničko ime dostupno, koristeći AJAX za slanje zahteva serveru bez osvežavanja stranice.

Potrebne tehnologije

- HTML za strukturu forme
- CSS za osnovno stilizovanje
- JavaScript za slanje AJAX zahteva
- Server u PHP-u (ili nekom drugom jeziku po izboru) za proveru dostupnosti korisničkog imena

1. HTML i CSS

Kreiramo formu (index.html) sa jednim poljem za unos korisničkog imena i mestom gde ćemo prikazati poruku o dostupnosti korisničkog imena.

```
<!DOCTYPE html>
<html lang="sr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Provera dostupnosti korisničkog imena</title>
</head>
<body>
    <h2>Provera dostupnosti korisničkog imena</h2>
    <form>
        <label for="username">Unesite korisničko ime:</label>
        <input type="text" id="username" name="username" onkeyup="checkAvailability()">
        <p id="message"></p>
    </form>

    <script src="script.js"></script>
</body>
</html>
```

Ovde imamo jedno polje za unos korisničkog imena i paragraf (<p>) gde ćemo prikazati poruku o dostupnosti imena. JavaScript kod je povezan preko fajla script.js, gde ćemo napisati AJAX logiku. Što se stila tiče, možemo dodati neke jednostavne naredbe za boje i naglašavanje poruka.

```
        /* Stil za poruke */
        #message {
            font-weight: bold;
        }
        .available {
            color: green;
        }
        .taken {
            color: red;
        }
```

2. JavaScript kod za AJAX zahtev

Sledeći korak je kreiranje JavaScript funkcije koja će slati AJAX zahtev serveru svaki put kada korisnik unese karakter u polje za korisničko ime.

```
function checkAvailability() {
    const username = document.getElementById("username").value;
    const message = document.getElementById("message");

    // Kreiramo novi AJAX zahtev
    const xhr = new XMLHttpRequest();
    xhr.open("POST", "check_username.php", true);
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

    // Kada stigne odgovor sa servera
    xhr.onload = function() {
        if (xhr.status === 200) {
            const response = xhr.responseText;
            if (response === "Korisničko ime je dostupno") {
                message.textContent = response;
                message.className = "available";
            } else {
                message.textContent = response;
                message.className = "taken";
            }
        } else {
            message.textContent = "Greška prilikom provere korisničkog imena.";
        }
    };

    // Šaljemo korisničko ime serveru
    xhr.send("username=" + encodeURIComponent(username));
}
```

U ovoj funkciji pokrivamo naredne operacije:

1. Preuzimamo vrednost polja **username** i kreiramo novi **XMLHttpRequest** objekat.
2. Podesimo metodu (POST), URL server skripte (check_username.php), i opcije zaglavlja.
3. Kada server odgovori, funkcija xhr.onload proverava status odgovora.
4. Ako je sve u redu (xhr.status === 200), obrađujemo odgovor.
5. Na osnovu odgovora (Korisničko ime je dostupno ili Korisničko ime je zauzeto), prikazujemo odgovarajuću poruku.

3. PHP server za proveru korisničkog imena

Konačno, potrebno je kreirati check_username.php skriptu koja proverava da li je korisničko ime zauzeto. Radi jednostavnosti, koristićemo statički definisanu listu zauzetih korisničkih imena (admin, korisnik i test su kao zauzeta imena).

Iako je PHP po planu i programu predviđen tek za sledeću vežbu, sada imate prilike da upoznate neke njegove osnovne komponente i vidite kako izgledaju bazične naredbe i funkcije. U narednim vežbama ćemo ovako proveravati da li je taj username već viđen negde u bazi podataka, ili ćemo izvršavati određena izračunavanja na osnovu podataka iz baze koji odgovaraju nekom kriterijumu.

```
<?php
// Lista zauzetih korisničkih imena
$zauzeta_imena = ["admin", "korisnik", "test"];

if (isset($_POST['username'])) {
    $username = $_POST['username'];

    // Proveravamo da li je korisničko ime zauzeto
    if (in_array($username, $zauzeta_imena)) {
        echo "Korisničko ime je zauzeto";
    } else {
        echo "Korisničko ime je dostupno";
    }
}
?>
```

Ovaj PHP kod radi sledeće:

1. Kreira niz $zauzeta_imena sa korisničkim imenima koja su već zauzeta.
2. Proverava da li je korisničko ime poslato preko POST metode.
3. Ako korisničko ime postoji u nizu $zauzeta_imena, server vraća poruku "Korisničko ime je zauzeto". Ako nije, vraća "Korisničko ime je dostupno".

Rezultat

Nakon što sve postavite, kada korisnik upiše korisničko ime u polje za unos, AJAX će slati zahtev serveru i prikazati poruku o tome da li je korisničko ime dostupno bez osvežavanja stranice.

Korisni resursi

1. Jednostavne AJAX vežbice: https://rua.ua.es/dspace/bitstream/10045/10699/4/Exercises.pdf
2. Detaljni tutorijali sa W3Schools: https://www.w3schools.com/js/js_ajax_examples.asp
3. Interaktivni kurs sa Univerzizeta u Vašingtonu gde možete koristiti strelice da pravite proges i prelazite njihova vežbanja: https://courses.cs.washington.edu/courses/cse154/18au/sections/04/slides/#/
4. Interesantno vežbanje koje koristi AJAX i YouTube klipove: https://codesandbox.io/p/sandbox/exercise-ajax-6n0tt