9. jQuery i AJAX

Univerzitet u Novom Sadu
Fakultet tehničkih nauka
Web programiranje – E2

jQuery

- JavaScript biblioteka koja olakšava pisanje klijentskih aplikacija
- Uključuje se u HTML stranice pomoću script taga
- Omogućuje jednostavan pristup elementima HTML stranice

jQuery

$ aktivira biblioteku jQuery

standardni JS kod:

```
document.addEventListener('ready', function() {....});
```

```
$(document).ready(function() {....});
```

JS kod sa korišćenjem jQuery-ja

jQuery

- Selekcija elemenata

- standardni JS kod:

```
let inputElem = document.getElementById(“ime”);
```

ili

```
let inputElem = document.querySelectorAll(“#ime”);
```

- JS kod sa korišćenjem jQuery-ja

```
let inputElem = $(“#ime”);
```

- Elemente selektujemo isto kao u CSS-u
- Vraćaju se jQuery wrapper objekti, koji olakšavaju korišćenje JS objekata koji predstavljaju elemente DOM stabla

jQuery

- Pristup poljima wrapper objekata
  - PRIMER: izvlačenje vrednosti ukucane u tekst-polje (input type text)

- standardni JS kod:

```
let inputElem = document.getElementById(“ime”);
alert(“Ukucali ste “ + inputElem.value);
```

- JS kod sa korišćenjem jQuery-ja

```
let inputElem = $(“#ime”);
alert(“Ukucali ste “ + inputElem.val());
```

jQuery

- Obrada događaja
  - Događaji mogu da se dodele bilo kojim selektovanim elementima
  - Imena događaja su isti kao u standardnom JavaScript-u

- standardni JS kod:

```
let divElem = document.querySelectorAll(“div”);
divElem.addEventListener(“click”, function() {
     let redElem = document.getElementByClassName(“red”);
     redElem.style.display = "none";
});
```

- JS kod sa korišćenjem jQuery-ja

```
$(“div”).click(function() {
     $(“.red”).hide();
});
```

AJAX

- Asynchronous JavaScript and XML
- Omogućuje razmenu podataka sa serverom bez da se osvežava čitava HTML stranica
- Osvežavaćemo samo deo stranice
- Razmena podataka je **asinhrona**
- Kombinovaćemo sve što smo prethodno naučili: slanje zahteva serveru (koji će biti asinhron) i izmenu HTML stranice manipulacijom DOM elemenata uz pomoć JavaScripta
- Mogu se koristiti različiti formati za razmenu, tj. server i klijent mogu da razmenjuju podatke u formatu **običnog teksta, HTML-a, XML-a i JSON-a**

AJAX

- Mi ćemo obično razmenjivati podatke u JSON formatu između klijenta i servera
- **Zašto JSON?**
  - Format nezavistan od tehnologije koju koristimo
  - Lak za razumevanje
  - Podaci su u formatu ključ: vrednost

```
{
  ime: “Pera”,
  prezime: “Peric”
}
```

```
[{
  ime: “Pera”,
  prezime: “Peric”
 },
 {
  ime: “Mika”,
  prezime: “Mikic”
 }
]
```

AJAX

- jQuery pojednostavljuje pravljenje AJAX poziva
  - primer **GET** zahteva

```
$("button").click(function(){
  $.get(”/user",
        function(data, status){
          alert("Data: " + data + "\nStatus: " +
                status);
        }
      );
});
```

podaci sa servera

status kod HTTP odgovora

link/putanja

callback

AJAX

- **GET** AJAX poziv ima dva parametra:
  - **Putanju** na serveru na koju upućujemo zahtev
  - **Callback** funkciju, koja će se izvršiti kada stigne odgovor sa servera
    - **ASINHRONA** OSOBINA: nema čekanja, funkcija se izvrši čim stigne odgovor

  - Callback funkcija ima dva parametra:
    - **data** – podaci koji stižu sa servera (mogu biti u svim formatima koje smo naveli, ali mi ćemo uglavnom koristiti podatke u JSON formatu)
    - **status** – status kod odgovora sa servera (npr. 200 OK)

  - Pošto je AJAX poziv asinhron, ako zaređamo više GET zahteva jedan ispod drugog, **ne znamo** koji će prvi završiti, tj. čija će se callback funkcija prva izvršiti

AJAX

- jQuery pojednostavljuje pravljenje AJAX poziva
  - primer **POST** zahteva

```
$("button").click(function(){
  $.post(”/user",
         {
          ime: ”Pera",
          prezime: ”Peric"
         },
         function(data, status){
            alert("Data: " + data + "\nStatus:
" + status);
  });
});
```

link/putanja

podaci koje šaljemo u okviru zahteva (JSON)

podaci sa servera

status kod HTTP odgovora

callback

AJAX

- **POST** AJAX poziv ima tri parametra:
  - **Putanju** na serveru na koju upućujemo zahtev
  - **Podatke** koje šaljemo na server (kod nas u primeru su u JSON formatu, njega ćemo najčešće koristiti)
  - **Callback** funkciju, koja će se izvršiti kada stigne odgovor sa servera

Razmena JSON-a

- Na serverskoj strani treba da imamo importovanu ***Jackson*** biblioteku i kreiran ***ObjectMapper***
- Prilikom izvršenja **GET** zahteva server treba da pošalje JSON
  - Server treba da zna kako da objekte napisane u Java programskom jeziku pretvori u JSON format
  - To nam omogućuje metoda ***writeValueAsString()*** klase ***ObjectMapper***, koja **serijalizuje** podatke iz Java formata (Java bean-ove) u JSON
- Prilikom izvršenja **POST** zahteva server treba da prihvati JSON podatke u zahtevu od klijenta
  - Server treba da zna kako da pročita podatke u JSON formatu i da ih pretvori u Java objekte
  - Ovo nam omogućuje metoda ***readValue()*** klase ***ObjectMapper***, koja deserijalizuje podatke iz JSON-a u Java bean-ove

Razmena JSON-a

- Na klijentskoj strani treba da imamo neke metode koje će moći isto da **serijalizuju** u JSON i **deserijalizuju** iz JSON-a kako bi uspešno bili kreirani JavaScript objekti
- Kada šaljemo GET zahtev putem AJAX poziva, na klijentskoj strani kada nam stigne odgovor sa servera treba da **deserijalizujemo** JSON uz pomoć metode ***JSON.parse(jsonString)***
- Kada šaljemo POST zahtev putem AJAX poziva, na klijentskoj strani treba da serijalizujemo podatke koje šaljemo u zahtevu serveru, uz pomoć metode ***JSON.stringify(obj)***