Dokumentacija - Šah, Računarski prakitkum 2

Dora Raštegorac, Vinko Lovrenčić i Luka Jandrijević

July 2021

1 rules.js, play.html

1.1 Varijable

Na početku koristimo angular kako bi stvorili zasebne "box"-ove odnosno šahovska polja. Pomoću toga sada imamo box-x-y gdje x i y označavaju koordinate tog polja.

Na početku `$(document).ready(function())`, na liniji 12, definiramo `chessPieces` u kojima se nalaze slike određenih figura. Slike figura prikazujemo HTML kodom. Tako na primjer string "&#9812" označava sliku bijelog kralja i tako dalje...

Varijabla `player` je zadužena za pratiti koji igrač je trenutno na redu, bijeli ili crni. Na početku je jednaka white jer je na početku bijeli na potezu. Varijabla `promotion` će nam pomoći da postavimo potrebne podatke pri promociji pijuna.

Pomoću varijable `select` pratimo može li se zadana figura micati, tip figure te pozicija figure, odnosno u kojem box-u se nalazi.

1.2 Postavljanje ploče

Na liniji 48, pozivamo funkciju koja je zadužena da polju s koordinatama i i j pridruži klasu `dark-box`, odnosno `light-box` ovisno o tome je li `i+j` paran ili neparan broj. Time dobivamo šahovnicu. S funkcijom `setNewBoard(box,i,j)` na polje sa koordinatama (i, j) postavljamo figuru ako se na početku igre tamo treba prikazati ta figura. Figuru postavljamo na zadano polje s funkcijom `setPiece(box,color,type)`. Na kraju još pozivamo funkciju `setTheme()` koja pomoču jQuery-a postavlja `light-box` i `dark-box` na zadanu boju.

1.3 Click event funkcije

Nakon postavljanje ploče definiramo funkciju koja je zadužena za click evente. Prvo obrađujemo slučaj ako je pijun promaknut. Odabirom od ponuđenih figura, ona se postavlja na potrebnu poziciju i postavlja klasu `placed`.

Kada kliknemo na polje tada se poziva funkcija zadužena za klik na klasu `box`. Ako je taj box, odnosno polje već od prije bilo odabrano to jest, ima klasu `selected`, tada se ta klasa za to polje briše i klasa `suggest` koja označava

moguća polja kojom se ta figura može kretati također briše, a varijablu `select` postavljamo na originalne vrijednosti.

Sljedeće što moramo provjeriti je slučaj ako se figura može micati. U tom slučaju znamo da se u `select.piece` nalaze sve potrebne informacije o figuri, boja i tip. Prvo provjerimo ako je odabrana druga figura iste boje. Ako je, tada odabiremo tu figuru. Figuru možemo pomaknuti samo na ona polja koja imaju klasu `suggest`. Ako taj box ima klasu `suggest`, tada pomoću funkcije `setPiece(box, color, type)` postavljamo tu figuru na zadano polje (box) i brišemo prethodno polje. Varijablu `select` opet vraćamo na originalne vrijednosti te mijenjamo igrača.

1.4 setPiece(box,color,type)

Funkcija `setPiece(box,color,type)` na početku provjerava je li figura "pojela" kralja na tom polju. Ako je, tada se poziva `showWinner()`, odnosno igra je završena. Dodatno, ona još provjerava je li pijun u poziciji za promaknuće. Varijabla `j` će nam reći na kojem rangu se nalazi figura. Ako je pijun u poziciji za promaknuće, tada se igra "zatamnjuje" pomoću `opacity` te pomoću jQuery-a prikazujemo popis figura u koje se pijun može promaknuti. Na kraju jednostavno postavimo HTML tekst u određeni HTML znak figure.

1.5 deleteBox(box)

Ova funkcija briše figuru sa određenog polja. Također briše klasu `placed` i `selected` sa određenog polja.

1.6 getNextMoves(selectedPiece, selectedBox)

Funkcijom `getNextMoves` dohvaćamo moguće poteze odabrane figure. Prvo dohvatimo informacije o figuri i spremimo ih u određene varijable. Zatim na sličan način dohvatimo i koordinate na kojima se ta figura nalazi. Ovisno o tome o kojoj figuri je riječ, pozivamo funkciju koja u varijablu `nextMoves` vraća sve moguće poteze te figure. Toj funkciji prosljeđujemo parametar `offset` koja predstavlja u kojim sve smjerovima se određena figura može micati.

Sastoji se od niza $[x, y]$ pri čemu x predstavlja pomak u okomitom smjeru dok y predstavlja pomak u vodoravnom smjeru. Tako na primjer top (eng. "rook") ima postavljen `offset` na $[[0, 1], [0, -1], [1, 0], [-1, 0]]$ jer se top može po pravilima micati gore,dolje,lijevo i desno. Za pijuna imamo u `offset`-u i $[0, 2]$ odnosno $[0, -2]$ jer kad je pijun na početnom položaju onda se može micati za 2 polja gore odnosno dolje, ovisno o tome o kojoj boji je riječ.

Za konja (eng. knight) `offset` ima niz $[2, -1]$ jer konj se može kretati za 2 polja gore i jedno polje ulijevo.Slično kao i za topa smo postavili i `offset` drugim varijablama.

1.7 pawnMoves(i, j, color, moves)

S ovom funkcijom dohvaćamo moguće poteze pijuna. U `tI` spremamo okomiti smjer a u varijablu `tJ` vodoravni smjer micanja figure. Prolazimo kroz varijablu `offset` te provjeravamo hoćemo li otići izvan granica ploče za odabranu figuru. Ako nećemo, tada provjeravamo još nalazi se na putu neka druga figura (to radimo pomoću klase placed). Sve te moguće poteze spremamo u `nextMoves` te ju nakraju funkcije vraćamo.

1.8 knightMoves(i, j, color, moves)

Ovom funkcijom dohvaćamo moguće poteze konja i kralja. Slično kao i prije, ako se na tom mjestu nalazi protivnikova figura tada ta figura može otići na to mjesto. To provjeravamo sa `indexOf(color)` koja vraća negativnu vrijednost ako ne pronađe index na kojem se nalazi podstring "color".

1.9 otherPiecesMoves(i, j, color, moves)

Ovom funkcijom dohvaćamo moguće poteze ostalih figura (kraljica, lovac, top). Ovdje koristimo varijablu `sugg` pomaže u detektiranju ako se neka figura nalazi na putu mogućih poteza. U tom slučaju gledamo je li ta figura protivnikova ili nije. Ako je protivnikova tada se i tamo možemo micati s tom figurom. U suprotnom, ne možemo jesti svoje figure pa se dalje ne možemo kretati.

1.10 suggestNextMoves

Funkcija pomoću koje za sve legalne poteze dodajemo tim poljima klasu `suggest`.

1.11 selectPiece(box)

Odabirom na figuru pozivamo ovu funkciju. Tom polju dodajemo klasu `selected` te u varijablu `select` postavljamo navedene vrijednosti. Dohvaćamo legalne poteze i sugeriramo na koja sva polja se figura može kretati.

2 Main page

U dokumentu *main2.html* nalazi se view glavne stranice web aplikacije.

Omogućili smo korisniku da u tražilicu upiše korisničko ime osobe protiv koje bi htio igrati i pošalje mu zahtjev. Ako ta druga osoba prihvati oboje su preusmjereni na stranicu sa igrom šah. (Vinkov dio)

Kako bi izbjegli stalno refreshanje stranice pri korisnikovom pretraživanju, listu osoba s tim ili sličnim korisničkim imenom dobivamo uz pomoć ajax-a kojeg pozivamo na dokument *main3.php*

Linije 28-31: naznačujemo da će se ta funkcija pozvati, tj.izvršiti samo kad korisnik pritisne gumb Search

Linije 34-58: funkcija u kojoj se nalazi ajax. Ako je uspješan u listu ću upisati element liste koji vraća stranica *main3.php*

Linije 62-72: imamo ponavljanje koda sa stranice za login - dio koji upućuje korisnika na pomoć pri učenju igranja šaha

Linija 78: link na stranicu za Logout koji smo u css-u oblikovali da izgleda kao gumb

2.1 style for main page

Linije 9-27: navodimo svojstva scrollbar-a kojeg ima/će imati lista korisnika

Linije 36-50: pozicioniramo naslov na stranici i navodimo njegova osnovna svojstva. Posebno, linije 46-49 zaobljuju bridove naslova - tu metodu koristimo i u oblikovanju ostalih div-ova

Linije 52-80: Definiramo izgled h1 , h2 i h3 tag-ova

Linije 82-90: pozicioniranje i definiranje diva u koji će stizati za poruku/obavijest - biti će hidden dok nema poruke

Linije 92-113: Oblikujemo kućicu za unos teksta kod pretraživanja

Linije 111-121: pozicioniramo kućicu za upis imena

Linije 124-132: pozicioniramo i oblikujemo gumb Search

Linije 134-149: oblikujemo i pozicioniramo div koji sadrži kućicu za pretraživanje imena

Linije 151-164: oblikujemo i pozicioniramo div koji sadrži listu korisničkih imena koja su pretražena

Linije 166-179: Oblikujemo listu i elemente liste

Linije 181-197: oblikuju div u kojem se nalaze linkovi

Linije 199-206: pozicioniranje linkova unutar div-a

Linije 208-232: oblikujemo link za Log out da izgleda kao gumb

(212-215: obikujemo rubove, 217-224: zadajemo veličine i pozicioniramo, 226-229: zaobljujemo rubove)

Od linije 234 uz pomoć @media naša stranica postaje responzivna. Uz svaki @media odredimo za koje veličine ekrana vrijedi i unutar njega unosimo izgled div-ova kojih želimo da se vide, za pojedine za koje ne želimo da se vide samo stavimo display:none;

2.2 main3.php

Sličan dokument ovom smo prošli na predavanjima. Jedina razlika je što u ovom prvo pristupamo bazi i iz nje izvlačimo potrebne podatke u polje te onda, ako polje nije prazno, vraćamo element liste koji sadrži ime osobe i gumb s kojim korisnik tu osobu zove na igru

3 style za login i register

Linije 15-29: oblikuju i pozicioniraju div u kojem se nalazi forma za login/register

Linije 31-46: oblikuju i pozicioniraju div u kojem se nalazi tekst i slika

Linije 48-56: obilikovanje h3 tag-a

Linije 58-66: pozicioniranje i oblikovanje slike šah

Linije 68-76: obilikovanje h2 tag-a

Linije 78-84: pozicioniranje forme za login/register

Linije 86-103: oblikovanje kućica u koje korisnik unosi podatke

Linije 103-110: oblikovanje forme

Linije 113-121: pozicioniranje gumba za login

Linije 123-142: pozicioniranje gumba za register

Linije 152-159: oblikovanje linkova za pomoć pri učenju

Linije 165-179: oblikovanje linkova

Od linije 187 uz pomoć @media naša stranica postaje responzivna. Uz svaki @media odredimo za koje veličine ekrana vrijedi i unutar njega unosimo izgled div-ova kojih želimo da se vide, za pojedine za koje ne želimo da se vide samo

stavimo display:none;

4 view, login/register i logout

4.1 chess.php

Početna stranica aplikacije je `chess.php` u kojem korisnik mora unijeti username i password kako bi nastavio dalje. `chess.php` prvo crta log-in formu te prati koji gumb je kliknut te ovisno o kliknutom gumbu prikazuje određenu stranicu. Ukoliko korisnik prvi put pristupa aplikaciji, tada se mora registrirati pritiskom na gumb register. Klikom na taj gumb korisnika se prosljeđuje na stranicu za registriranje u kojem je potrebno upisati željeni username i password te svoju email adresu. Ako je korisnik krivo utipkao email adresu o tome se javlja greška. Ako username već postoji tada se opet javlja greška i ispisuje poruka.

4.2 login

U `login.php` imamo dvije funkcije: `procesiraj_login()` i `procesiraj_novi()`. Prva funkcija provjerava da li se u varijabli `POST` nalazi username i je li taj username valjan. Zatim iz baze podataka dohvaća taj username te provjerava je li se hashirane šifre podudaraju. Ako je sve dobro tada otvara novu sesiju, u tu sesiju sprema username korisnika te ga prosljeđuje na iduću stranicu. Funkcija `procesiraj_novi()` se poziva pri registraciji. Ona služi da kada se korisnik ide registrirati tada u bazu podataka sprema tog korisnika sa unesenim usernameom i passwordom.

4.3 register.php, processRegistration.php

U `register.php` se nalazi funkcija koja služi za crtanje forme pri registraciji korisnika. Nju pozivamo u `processRegistration.php` kada korisnik još nije kliknuo na gumb register. Jednom kada je kliknuo gumb register, poziva se funkcija `procesiraj_novi()` iz `login.php`.

5 Baza podataka

5.1 users

U bazi podataka imamo nekoliko tablica. Prva tablica je `users` u kojima spremamo podatke poput: `username` koji je ujedno i primarni ključ jer nije dozvoljeno da postoje dva korisnika s istim username-om, `opponet` u kojem spremamo username protivnika protiv kojeg korisnik igra (na početku je prazan), `color` koja označava s kojom bojom korisnik igra (black ili white), `gameId` koji označava u kojoj partiji šaha korisnik igra (ako je `gameId` jednak 0 to znači da korisnik ne igra trenutno partiju). Na kraju, još moramo čuvati i hashiranu lozinku korisnika koju provjeravamo tijekom logiranja na stranicu.

6 Model

6.1 User

U modelu imamo nekoliko klasa. Najbitnija od njih svih je klasa `User`. U toj klasi imamo definirane funkcije kao što su `getAllUsers()` pomoću koje dohvaćamo sve korisnike aplikacije. Pomoću funkcije `getUserByUsername(username)` dohvaćamo traženog korisnika iz baze podataka. Funkcijom `availablePlayers` pronalazimo korisnike koji trenutno ne igraju partiju, odnosno njima možemo poslati zahtjev za igru. Znamo da su određeni korisnici slobodni za partiju ako im je `gameId` = 0, inače ako je korisnik u partiji tada mu je gameId drugačiji broj. Na kraju, imamo funkciju `deleteGame(gameId)` koja služi da obriše `gameId`, odnosno da taj `gameId` postavi ponovno na 0 i tako označi da su ta dva igrača opet slobodni za igru.

6.2 Game

U klasi `Game` imamo funkciju `findLastMove` koja pronalazi zadnji potez igre `gameId`. Pomoću funkcije `findColor` možemo pronaći s kojom bojom korisnik igra. Funkcija `insert` ubacuje potez u bazu podataka gdje je `gameId`=`id`