**SVEUČILIŠTE U ZAGREBU, FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

36. Chat

**REACT.JS**

Bruna Anđelić

**Zagreb, prosinac 2016.**

# Uvod

U ovom će se izvješću obrađivati tema React.js koji je danas sve popularnija i korištenija JavaScript programska knjižnica. Kroz ovo izvješće u idućih nekoliko poglavlja opisat će se ukratko što je to React.js i koja je glavna razlika između njega i ostalih JavaScript programskih knjižnica. Nadalje objasniti će se kratka povijest React.js-a. Kako se uopće došlo do ideje za React.js, tko ga je razvio te kako se i danas ravija. Također budući da je React.js prilično velika knjižnica sa mnoštvom različitih mogućnosti te koncepata koji se koriste u jednom će se poglavlju objasniti one najosnovnije stvari vezane uz programiranje u React.js-u. Također uz React.js veže se i posebni arhitekturni obrazac koji se uz React.js najviše koristi, a to je Flux koji će također biti pobliže objašnjen u jednom poglavlju.

# Što je React.js?

React.js je *open-source* JavaScript programska knjižnica koja se koristi za *render* HTML podataka. Korištenjem React.js-a stvaramo interaktivne komponente sa stanjem(*statefull*) koje se mogu ponovno iskoristiti. Jedna od važnih prednosti React.js-a je činjenica da komponenta vraća izravno kao HTML pa se tako isti kod može pokretati i na poslužitelju (*server-side rendering*) i na klijent(*client-side rendering*), ali i izomorfno kombiniranjem najboljem iz oba pristupa.

Jedna od glavnih karakteristika React.js-a je činjenica da on predstavlja samo *view* sloj MVC arhitekture i to je jedna od glavnih razlika između React.js-a i ostalih JavaScript knjižnica kao što su Angular ili Backbone koji predstavljaju potpunu arhitekturu uključujući i *view* sloj. Iz tog se razloga ne može izgraditi potpuno funkcionalna dinamička aplikacija korištenjem isključivo React.js-a. Iako se iz ove usporedbe može zaključiti da se druge JavaScript knjižnice korisnije i bolje za koristiti to nije tako. Teško ih je uspoređivati direktno jer rade na drugim principima tako da se ne može reći da se za React.js ne može reći da je bolji ili gori od drugih, ali zbog svojih karakteristika koje će se opisati u sljedećim podpoglavljima postaje sve popularnija i sve korištenija JavaScript programska knjižnica.

Osnovna karakteristika React.js-a je jednosmjerni protok podataka. Svojstva(*properties*), koja su skup nepromjenjivih (*immutable*) vrijednosti prenose se između komponenti kao svojstva unutar HTML[[1]](#footnote-1) oznaka. Upravo su svojstva(*properties, props*) zajedno sa stanjem(*state*) jedini način komunikacije među komponentama, a komponente su osnovna jedinica koju React.js koristi. Budući da se upravo na komponentima, svojstvima i stanju temelji React.js te da se ti koncepti najbitniji prilikom programiranja u React.js-u oni su pojedinačno opisani u 4. poglavlju. Najvažnije je za neki opće pregled shvatiti da se sva svojstva(*props*) mogu prenositi samo u jednom smjeru. Komponenta ne može direktno mijenjati niti jednu vrijednost svojstva koja joj je proslijeđena, ali može proslijediti *callback* funkciju koja će modificirati vrijednost. Taj mehanizam koji React.js koristi poznat je pod nazivom „*properties flow down, actions flow up*“ i on možda najslikovitije opisuje mehanizam. Dakle vrijednosti svojstava(*props*) prenose se u jednom smjeru, ali se *callback* funkcijom akcije prenose u smjeru suprotnom smjeru protoka podataka.

Nadalje React.js koristi virtualni DOM[[2]](#footnote-2) koji omogućuje selektivni *render* komponenata baziran na promjenama stanja. React kreira *in-memory* strukturu podataka(*cache*) i određuje razlike između dosadašnjem i novog DOM-a te ažurira(*update*) potrebne razlike i na taj se način DOM učinkovito prikazuje. Ovim pristupom programer može pisati kao da će se cijela stranica prevesti ponovno na svaku promjenu podataka dok React zapravo ažurira u DOM-u samo dijelove komponenti koje se zapravo mijenjaju. Također komponente se u React.js-u pišu u JSX-u. On dozvoljava navođenje HTML-a korištenjem HTML oznaka kako. JSX je statični, objektno orijentirani programski jezik dizajniran kako bi se izvodio u modernim web preglednicima. Bitno je napomenuti i da je osnovna arhitektura React.js aplikacije nadilazi samo prikazivanje HTML-a u web preglednicima te uvodi dinamičnost.

# Povijest React.js-a

React.js razvio je softver inženjer iz Facebooka[[3]](#footnote-3), Jordan Walke. Osnovi utjecaj na njegov rad bio je XHP, PHP razvojni okvir(*framework*) za HTML komponente. U vrijeme kada ga je razvijao, Jordan Walke radio je na dijelu za oglase, ali se prebacio u „*Product Infra team*“ kako bi se mogao *full-time* posvetiti razvoju React.js-a. Prilikom razvoja prva stvar na kojoj je testirao React bila je React verzija za „*lajkanje“* i komentiranje. To je bio prvi korak u prevođenju Facebooka na React.js.

Do te prve uporabe React.js-a Instagram[[4]](#footnote-4) se pripojio Facebook-u te je bivši inženjer Facebook tima za fotografije, Pete Hunt, prešao u Instagram tim te napravio prvu verziju Instagram obavijesti za web koristeći upravo React. Budući da je kod za Instagram bio potpuno odvojen od Facebook koda zatražio je da se React izdvoji iz Facebook koda kako bi ga se moglo koristiti i za razvoj Instagrama. Tako gledajući može se reći da je Instagram prvi „vanjski“ korisnik React.js-a. Taj je potez bio začetnik ideje za „*open-source*“ React.js-a koji je omogućio da ga danas može koristiti cijeli svijet.

Prvo korištenje React.js-a u sklopu Facebooka bilo je 2011.godine dok se prvi puta za Instagram iskoristio 2012.godine. *Open-source* je postao na JSConf US u svibnju 2013.godine.

Danas je React.js jedan od najpopularnijih repozitorija na Git-u te danas ima 876 suradnika(„*contributor*a“)[[5]](#footnote-5). Broj stranica koje koriste React.js povećava se iz dana u dan, a neke od popularnijih su: BBC[[6]](#footnote-6), eBay, Instagram, Facebook, IMDb[[7]](#footnote-7), Imgur[[8]](#footnote-8) i mnoge druge. Popis web stranica napisanih u React.js-u dostupan je putem poveznice „*https://github.com/facebook/react/wiki/Sites-Using-React“*.

Također je važno napomenuti da se iz React.js-a razvio i React Native koji omogućuje stvaranje *nativnih* mobilnih aplikacija korištenjem JavaScripta i React.js-a te tako omogućuje razvoj iOS, Android i UWP[[9]](#footnote-9) aplikacija. React Native prvi puta je predstavljen na „*React.js Conf*“ u veljači 2015. godine, a postao je *open-source* u svibnju iste godine.

# Osnovni dijelovi React.js programskom koda

Kao što je prija napomenuto osnovna jedinica kojom se gradi Reac.js aplikacija naziva se komponenta. Svaka komponenta ima stanje(*state*) i svojstva(*props*). Stanje se postavlja i mijenja korištenjem posebne metode „*setState*“ koja ja okidač(*trigger*) za ažuriranje(*update*) UI-a[[10]](#footnote-10). Kao što smo naglasili DOM se ažurira selektivno ovisno o promjenama u stanju. Svojstva se koriste za definiranje komponenti korištenjem atributa *props* koji su unutar komponenta dostupni uporabom *this.props* i mogu se koristiti u *render* metodi kako bi se prevodio dinamički sadržaj.

Render metoda je jedini dio koji je obavezano dodati kako bi se kreirala nova komponenta. Unutar metode render potrebno je imati *return* koji vraća dio koda koji vizualno predstavlja komponentu kojoj je pridružen. Uz render metodu mogu se koristiti i *lifecycle* metode koji su jako korisne kada se od komponente traži da nešto napravi. Glavne takve metode su : *componentWillMount, componentDidMount, shouldComponentUpdate* i *componentWillUnmount*. Takve metode govore nam kada je komponenta u jednom od stavnja. Primjerice *componentWillMount* poziva se jednom na klijentskoj i serverskoj strani prije nego što će se pozvati metoda render, dok se metoda *componentDidMount* poziva jednom samo na klijentskoj strani nakon što se izvrši metoda render. Nadalje metoda *shouldComponenetUpdate* vraća vrijednost koja govori o tome treba li se ili ne komponenta ažurirati, a metoda *componentWillUnmount* poziva se prije nego što će se komponenta *unmountati.* Korištenjem ovih metoda točno se zna u kojem će se trenutku pozvati kod koji im je pridružen što može biti jako korisno i praktično.

Rukovanje događajima korištenjem React.js-a vrlo je slično rukovanjem događajima nad DOM elementima uz iznimku malo sintaksne razlike. React događaji imenuju se korištenjem *camelCase*, a ne *lowecase* te se pomoću JSX-a funkcije prenose kao *event handler*, a ne kao *string*.

U poglavlju su opisana neka od glavnih dijelova koda, ali React.js ima mnoštvo različitih dijelova i mogućnost pa su i neke od osnovnih koncepata nekom tko nikada nije koristio React.js prilično apstraktne. U poglavlju se samo pokušalo navesti osnovne ideje i dijelove koda. Najbitnije je shvatiti da se komponenta uvijek sastoji u nekom stanju te da se ovisno o stanju ona može mijenjati te da se osim stanja informacije prenose *propsima*. Ostale se karakteristike koda pisanog React.js-om može lako naučiti proučavanjem službene Facebook dokumentacije[[11]](#footnote-11) koja pobliže objašnjava svaku od komponenata koda.

# Što je Flux?

Flux je arhitekturni obrazac koju Facebook koristi prilikom rada s React.js-om. Važno je napomenuti da to nije niti *framework* niti *library*, već arhitekturni obrazac koja se slaže s React.js-om i konceptom jednosmjernog protoka podataka. Flux arhitekturni obrazac može se implementirati na više načina, a Redux[[12]](#footnote-12) implementacija koristi se u Facebook-u te je najkorištenije implementacija tog obrasca.

Glavna ideja je, kao što je napomenuto jednosmjerni protok podataka. Jedan uobičajeni slučaj korištenja bio bi da korisnik pokreće akciju(*action*) koja zatim šalje podatke u *reducer* koji služi kao sabirna točka svih ackija. Svaki *reducer* na osnovu akcije i trenutnog stanja aplikacije stvara novo stanje. Bitno je naglasiti dodatno činjenicu da se stvara novo stanje, dakle nema modifikacije trenutnog stanja. To se novo stanje šalje u spremište(*store*). Spremište je zaduženo za prijenos novog pogleda korisniku i za aktivaciju nove akcije kada je to potrebno.

Usporedimo li Redux sa arhitekturom MVC vidimo da je jednostavnija, osnovno izbjegava neke složene konstrukcije, ali također ima laku predvidivost. Kao što je naznačeno cijeli je fokus stavljen na trenutno sranje te je samo to bitno. Budući da je protok podataka jednosmjeran lako se predviđaju pogreške te se aplikacije na taj način brže razvijaju.

# Zaključak

React.js jedna je od najmlađih programskih knjižnica JavaScripta te se kao takva još uvijek intenzivno mijenja. U ovom su izvješću nabrojane su mnoge karakteristike React.js-a koje ga danas čine sve više popularnim i korištenim u svijetu web developmenta, ali kada se odluči koristiti React.js potrebno je shvatiti da je se on još uvijek dinamički mijenja te da je potrebno pratiti novosti iz React.js svijeta te pratiti bilo kakve promjene u dokumentaciji kako ne bi imali problema prilikom nekih većih promjena kao što su izbacivanje neke od funkcionalnosti i sl. Kod spomenutom React Native-a situacija je još gora jer je on još mlađi. Ali usprkos spomenutim manama sama činjenica da je React.js toliko raširen i korišten govori o tome da se isplati koristiti ga usprkos manama. Upravo smo se zato i mi u ovom projektu odlučili koristiti ga jer se kod projekta kao što je chat treba dobiti velika brzina osvježavanja sadržaja ekrana koji je vrlo dinamičan.

1. **HTML** – HyperText Markup Language [↑](#footnote-ref-1)
2. **DOM** - The Docuemnt Object Model [↑](#footnote-ref-2)
3. www.facebook.com [↑](#footnote-ref-3)
4. www.instagram.com [↑](#footnote-ref-4)
5. Podatak preuzet sa službene git stranice : ***https://github.com/facebook/react*** [↑](#footnote-ref-5)
6. http://www.bbc.com/ [↑](#footnote-ref-6)
7. http://www.imdb.com/ [↑](#footnote-ref-7)
8. http://imgur.com/ [↑](#footnote-ref-8)
9. Universal Windows Platform [↑](#footnote-ref-9)
10. User interface [↑](#footnote-ref-10)
11. https://facebook.github.io/react/docs/ hello-world.html [↑](#footnote-ref-11)
12. https://github.com/reactjs/redux [↑](#footnote-ref-12)