Uvod u svet JavaScript aplikacija

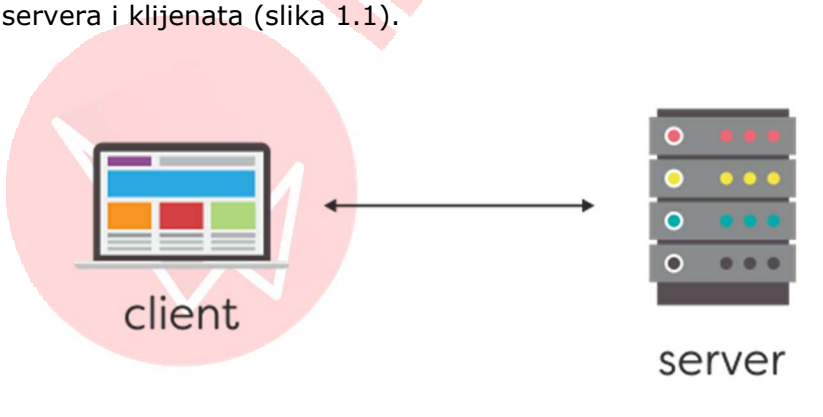
Tokom ranih godina razvoja weba, web sajtovi su po svojim sposobnostima bili vrlo ograničeni, pa su tako korisnicima prevashodno omogućavali pregled tekstualnih podataka propraćenih ponekom slikom, grafikom ili ilustracijom. Ipak, s vremenom su, s razvojem tehnologije, a prevashodno kompjuterskih mreža i web pregledača, otvarane mogućnosti da web sajtovi postanu moćniji i kompleksniji, pa da samim tim i korisnicima obezbede mnogo viši stepen interaktivnosti. Tako su web sajtovi po svom izgledu, ali i po načinu funkcionisanja, s vremenom počeli sve više da podsećaju na desktop aplikacije.

Danas je web jedno od primarnih okruženja za koje se razvija moderni aplikativni softver, a čak i jednostavni web sajtovi poseduju barem poneku osobinu prave web aplikacije. S obzirom na to da je predmet našeg interesovanja frontend programiranje, u kursu pred vama bavićemo se razvojem web aplikacija, sa posebnim akcentom na JavaScript jeziku.

Šta su web aplikacije?

Web sajtovi sa osobinama kompjuterskih programa drugačije se nazivaju web aplikacije. U funkcionisanju web aplikacija učestvuje određena programska logika, napisana nekim od programskih jezika. Email klijent *Gmail*, servis za pregled geografskih mapa *Google Maps* ili društvena mreža *Facebook* neki su od klasičnih primera web aplikacija. Upotreba različitih programskih jezika za kreiranje upravo spomenutih web aplikacija njima omogućava da svojim korisnicima ponude napredne slučajeve korišćenja, koji daleko prevazilaze okvire jednostavnog prezentovanja podataka.

Web aplikacije postoje u specifičnom okruženju u okviru internet servisa koji se naziva World Wide Web. Tako je okruženje web aplikacija uslovljeno postojanjem mreže i dva tipa kompjutera – servera i klijenata (slika 1.1).



Slika 1.1. Klijent-server arhitektura

Serveri i klijenti na webu komuniciraju uz poštovanje pravila HTTP protokola, pa se zbog toga oni veoma često drugačije nazivaju HTTP klijenti i HTTP serveri.

S obzirom na to da je okruženje u kome postoje web aplikacije gotovo uvek sačinjeno iz minimalno dva kompjutera, to praktično znači da postoje i dva različita mesta na kojima je moguće izvršiti određenu programsku logiku:

- klijentski deo (engl. *frontend*),
- serverski deo (engl. *backend*).

Klijentski deo web aplikacija izvršava se unutar web pregledača, a jedini programski jezik koji web pregledači razumeju jeste JavaScript.

Serverski deo web aplikacija postoji i izvršava se unutar kompjutera koji se drugačije nazivaju serveri. Programska logika web aplikacija na serveru može biti formulisana korišćenjem različitih programskih jezika – PHP, C#, Java, Python…

Na kraju se može rezimirati da web aplikacije podrazumevaju dve različite pozornice za izvršavanje programske logike:

- klijentsku, na kojoj se isključivo koristi JavaScript, i
- serversku, na kojoj je moguće koristiti brojne backend jezike.

Pitanje

U svetu weba i HTTP protokola, najčešći tip HTTP klijenata su:

a) web pregledači
b) web serveri
c) poslovne aplikacije
d) aplikacije za obradu teksta

Objašnjenje:

Klijentski deo web aplikacija izvršava se unutar web pregledača. Tako su web pregledači (Chrome, Firefox, Safari, Edge…) najčešći oblik HTTP, odnosno web klijenata.

Šta su JavaScript aplikacije?

JavaScript aplikacije su web aplikacije za čije kreiranje se u većoj ili manjoj meri koristi JavaScript jezik. Danas je praktično bilo koja aplikacija na webu ujedno i JavaScript aplikacija. Drugim rečima, gotovo da ne postoji web aplikacija koja u većoj ili manjoj meri ne koristi JavaScript jezik. Ipak, količina logike koja se realizuje upotrebom JavaScript koda može znatno varirati od aplikacije do aplikacije.

JavaScript se može koristiti za realizaciju brojnih aspekata web aplikacija, od klijentske validacije i generisanja prezentacione logike pa sve do korišćenja JavaScripta i na serveru, za formulisanje pozadinske logike. Tako je, pored nešto ranije spomenutih popularnih backend jezika, i JavaScript jedan od jezika koje je moguće koristiti na serveru, upotrebom izvršnog okruženja Node.js. Na taj način je moguće kreirati JavaScript aplikacije u punom značenju te reči, odnosno web aplikacije čija je kompletna programska logika, kako na klijentu tako i na serveru, realizovana korišćenjem JavaScript jezika.

Stoga možemo zaključiti da se pojam JavaScript aplikacija može odnositi na:

- web aplikacije koje su u potpunosti realizovane korišćenjem JavaScript jezika, na klijentu, ali i na serveru, i na
- web aplikacije kod kojih se u okviru prezentacionog sloja JavaScript koristi za obavljanje jednog dela aplikativne logike.

Kako se kreiraju JavaScript aplikacije?

JavaScript aplikacije moguće je kreirati na dva načina:

- korišćenjem kombinacije JavaScript jezika i različitih Web API-ja, čije korišćenje obezbeđuju web pregledači i
- korišćenjem različitih biblioteka (Vue, React, Angular…) uz opciono korišćenje izvršnog okruženja Node.js, npm menadžera paketa i različitih alata koji olakšavaju i ubrzavaju razvoj.

U ovom kursu prvo će biti ilustrovano kreiranje jedne JavaScript aplikacije isključivo korišćenjem JavaScript jezika i različitih Web API-ja. U nastavku kursa kreiranje takve aplikacije biće demonstrirano i upotrebom nekoliko popularnih biblioteka ili softverskih okvira namenjenih frontend programiranju.

Web API-ji za kreiranje JavaScript aplikacija

U prethodnim redovima je rečeno da osnovni način za kreiranje JavaScript aplikacija podrazumeva korišćenje JavaScript jezika i različitih funkcionalnosti koje web pregledači izlažu na korišćenje. Skupovi funkcionalnosti koje web pregledači stavljaju na raspolaganje programeru drugačije se nazivaju aplikativni programski interfejsi weba, odnosno skraćeno Web API-ji. Moderni web pregledači izlažu veliki broj različitih API-ja koje je moguće koristiti prilikom klijentskog programiranja na webu. U nastavku će biti navedeni najznačajniji Web API-ji koji se aktivno koriste prilikom razvoja JavaScript aplikacija.

DOM API

Svakako najznačajniji Web API jeste onaj koji omogućava programabilnu intervenciju nad strukturom i stilizacijom dokumenta. Reč je o Web API-ju koji se naziva DOM API. DOM je skraćenica koja označava pojam *Document Object Model,* što je zapravo objektna reprezentacija jednog HTML dokumenta.

Objektna reprezentacija strukture dokumenta omogućava JavaScript jeziku da na lak način vrši interakciju sa dokumentom. Tako je DOM API osnovni alat koji se prilikom razvoja JavaScript aplikacija koristi za parcijalnu izmenu sadržaja, kreiranje, izmenu i brisanje elemenata i manipulaciju stilizacijom.

XMLHttpRequest i Fetch API

Osnovni način funkcionisanja HTTP protokola oduvek je bio jedan od glavnih otežavajućih faktora za razvoj JavaScript aplikacija koje izgledaju i ponašaju se kao prave native aplikacije. Naime, HTTP se zasniva na međusobnom smenjivanju zahteva i odgovora, pri čemu podrazumevano svaki serverski odgovor sadrži i potpuno osvežavanje stranice unutar klijentskog pregledača, a vrlo često i preusmeravanje na potpuno novu stranicu.

Ipak, web pregledači izlažu skup funkcionalnosti koje je moguće koristiti za slanje HTTP zahteva korišćenjem JavaScript koda koji mi samostalno pišemo. Takvi pristupi omogućavaju da se HTTP zahtevi pošalju bez potrebe za osvežavanjem kompletne stranice – odnosno da se obavi slanje HTTP zahteva koje je potpuno nevidljivo za korisnika koji pregleda web stranicu. Na taj način otvaraju se mogućnosti za moćnu manipulaciju sadržajem i strukturom dokumenta, koji je moguće ažurirati parcijalno u zavisnosti od potreba i namene web sajta. Skup funkcionalnosti koji takvo nešto omogućava drugačije se naziva AJAX (***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML*), a centralne figure takvog modela su dva posebna Web API-ja: XMLHttpRequest i Fetch API.

XMLHttpRequest i Fetch API-ji omogućavaju komunikaciju sa serverom. Takva komunikacija ogleda se u mogućnosti slanja i primanja podataka u JSON, XML, HTML ili čistom tekstualnom obliku. Prilikom razvoja JavaScript aplikacija, ova dva skupa funkcionalnosti se mogu koristiti za slanje i prijem podataka sa servera, čime se otvara mogućnost da se podaci JavaScript aplikacija pouzdano i trajno sačuvaju na serveru.

History API i Location API

U samu srž weba i HTTP protokola utkan je i pojam stranica. Tako su web sajtovi tipično sačinjeni iz većeg broja zasebnih stranica koje se nalaze na različitim URL adresama, odnosno na različitim putanjama jednog domena. Takva strukturiranost još jedna je osobina koja web aplikacije distancira od native aplikacija, pogotovu ukoliko je prelazak sa stranice na stranicu propraćen potpunim osvežavanjem prikaza unutar web pregledača. Na sreću, potpuno osvežavanje stranice se može predupredi korišćenjem Web API-ja iz prethodnog poglavlja, ali i dalje preostaje problem rukovanja putanjama i istorijom pretrage. Web pregledači za rešavanje takvog problema izlažu dva API-ja: History API i Location API.

Location API omogućava da se na klijentu dobiju informacije o trenutnoj URL adresi, ali i da se obavi osvežavanje ili preusmeravanje na neku novu putanju. S obzirom na to da web pregledači beleže istoriju posećenih stranica i omogućavaju kretanje kroz takvu istoriju (korišćenjem komandi *Back* i *Forward*), rukovanje takvom istorijom još jedan je od važnih aspekata razvoja JavaScript aplikacija. History API obezbeđuje funkcionalnosti koje omogućavaju pristup istoriji posećenih stranica za trenutnu sesiju web pregledača.

History API i Location API veoma često se koriste za razvoj jednostraničnih aplikacija (engl. *SPA*), što je jedan od tipova modernih web aplikacija o kome će biti više reči u nastavku ove lekcije.

Web Storage API i IndexedDB

JavaScript aplikacije veoma često imaju potrebu da unutar web pregledača sačuvaju određene podatke koji će se koristiti isključivo na klijentu. Za obavljanje takvog posla web pregledači obezbeđuju nekoliko mehanizama, koji se koriste u zavisnosti od potreba aplikacije i kompleksnosti podataka.

Za čuvanje jednostavnijih podataka, prilikom razvoja JavaScript aplikacija može se koristiti Web Storage API. Reč je zapravo o skupu funkcionalnosti koje omogućavaju čuvanje podataka unutar dva tipa skladišta:

- Session Storage – skladište koje omogućava čuvanje podataka tokom jedne sesije web pregledača i
- Local Storage – skladište koje omogućava da se podaci unutar pregledača sačuvaju čak i kada se web pregledač zatvori.

Za kompleksnije scenarije čuvanja podataka na klijentu web pregledači obezbeđuju i bazu podataka koja se naziva IndexedDB. IndexedDB obezbeđuje brojne pogodnosti, koje mogu biti korisne ukoliko aplikacija barata složenim strukturama podataka – univerzalnost tipova, transakcije, indekse i mnogo veći raspoloživi prostor za skladištenje.

Service Worker API

Service Worker API je skup funkcionalnosti pomoću kojih je moguće kreirati JavaScript logiku koja se ponaša kao posrednik između web aplikacije, odnosno web pregledača i mrežne konekcije. Tako je u pitanju tehnologija koju je moguće koristiti za kreiranje logike koja može da kontroliše web aplikaciju, presretanjem HTTP zahteva koji se razmenjuju između web pregledača i web servera.

Logika Service Worker API-ja definiše se u zasebnim JavaScript fajlovima, a zatim se sa servera preuzima i smešta unutar web pregledača. Takav kod izvršava se unutar zasebne niti, nezavisno od ostatka JavaScript koda aplikacije za koju je vezan. Service Worker API jedan je od osnovnih skupova funkcionalnosti koji omogućava rukovanje situacijama u kojima ne postoji mrežna konekcija, što aplikacijama omogućava da funkcionišu u nepovezanom režimu.

Tipovi web aplikacija

Nešto ranije je rečeno da pojam JavaScript aplikacija može podrazumevati programsku logiku koja je u potpunosti (i na serveru i na klijentu) realizovana upotrebom JavaScript jezika ili JavaScript programsku logiku samo na klijentskom delu. Stoga nije teško zaključiti da je pojam JavaScript aplikacija veoma širok i da se može odnositi na aplikacije koje poseduju znatno drugačiju unutrašnju strukturu. Zato ćemo se u nastavku lekcije upoznati sa nekim od osnovnih tipova web aplikacija koje se danas u većoj ili manjoj meri realizuju korišćenjem JavaScript jezika. U pitanju su sledeći tipovi aplikacija:

- Multi-page Applications,
- Single-page Applications,
- Isomorphic (Universal) Applications,
- Progressive Web Applications.

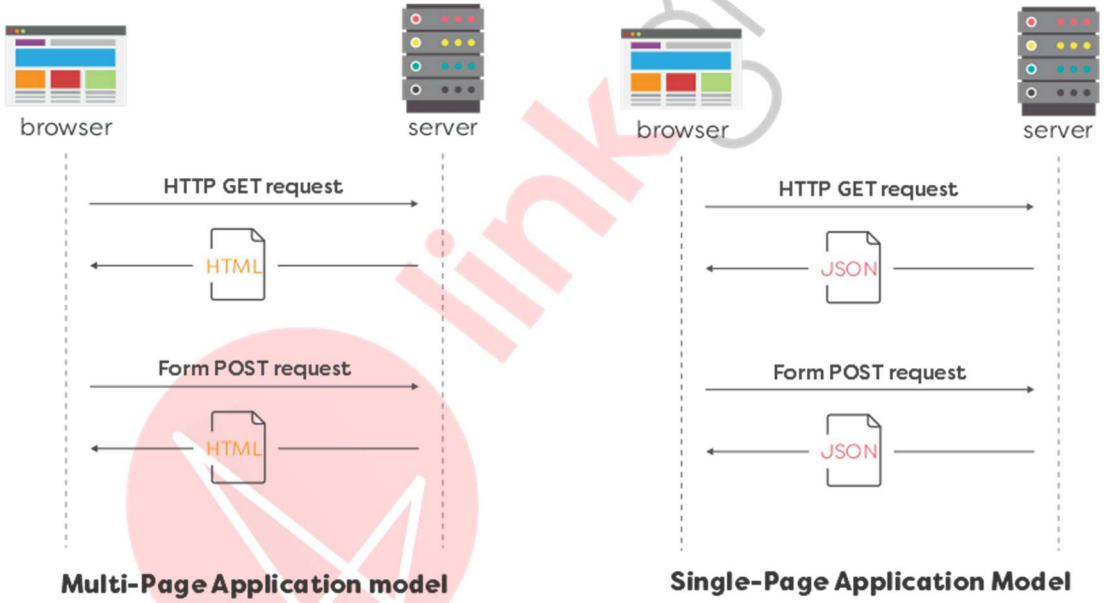
Poznavanje osobina upravo navedenih tipova web aplikacija pomoći će vam da na lakši način uvidite sve one aspekte takvih aplikacija koji se realizuju korišćenjem JavaScript jezika.

Jednostranične i višestranične web aplikacije

Prva klasifikacija koju je potrebno razumeti jeste ona koja web aplikacije deli na višestranične (engl. *multi-page*) i jednostranične (engl. *single-page*). Osnovna razlika između ova dva tipa aplikacija odnosi se na tip sadržaja koji se dobija od servera, kao i na način na koji web pregledač i klijentska logika rukuju URL adresama.

Višestranične aplikacije (MPA) funkcionišu po tradicionalnom modelu, koji podrazumeva da svaki klijentski zahtev rezultira isporukom kompletne HTML stranice. Pri tome je prelazak sa jedne na drugu stranicu propraćen ponovnim učitavanjem kompletnog prikaza unutar web pregledača. Sa druge strane, kod Single-page aplikacija stranica se nikada ne osvežava u potpunosti. Prvim otvaranjem Single-page aplikacije web pregledaču se isporučuje sva potrebna klijentska logika koja je zadužena za komunikaciju sa serverom i za generisanje prezentacije. Tako Single-page aplikacije umesto HTML stranica dobijaju podatke u JSON ili XML obliku, koje klijentska, JavaScript logika koristi za generisanje prikaza. Na kraju korisnik ima utisak da koristi aplikaciju koja po svom načinu funkcionisanja podseća na native aplikacije.

Osnovna razlika između tradicionalnih (MPA) i Single-Page aplikacija (SPA) ilustrovana je slikom 1.2.



Slika 1.2. Uporedni prikaz uprošćenog funkcionisanja MPA i SPA

Na levoj polovini slike 1.2. je moguće videti tradicionalni model funkcionisanja web aplikacija. Klijent upućuje zahtev, server u kombinaciji sa pozadinskom logikom takav zahtev obrađuje i generiše HTML kod koji isporučuje klijentu. Prilikom svakog novog klijentskog zahteva, takav postupak se ponavlja – prezentacija se generiše na serveru i kao zaokružena celina isporučuje klijentu kome preostaje samo da takav sadržaj parsira i prikaže.

Na desnoj polovini slike 1.2. ilustrovana je osnovna osobina Single-page aplikacija. Možete videti da se prezentacija (HTML) ne generiše na serveru, već da se klijentu isporučuju podaci, najčešće u JSON ili XML formatu (mada JSON danas ipak ima primat). Klijent zatim, na osnovu takvih podataka, samostalno formira prezentaciju. Tako JavaScript u Single-page modelu efikasno preuzima deo logike koja se tiče rukovanja prezentacijom.

U priči o višestraničnim i jednostaničnim aplikacijama potrebno je razumeti i nekoliko činjenica koje se često pogrešno interpretiraju. MPA se mogu realizovati i bez JavaScript jezika, ali programera ništa ne sprečava da i za njihov razvoj koristi JavaScript. Razlika je samo u obliku serverskog odgovora i načinu na koji se takav odgovor obrađuje.

Još jedan mit se odnosi na to da Single-page aplikacije ne mogu biti podeljene na više stranica. Naime, i Single-page aplikacije konceptualno mogu biti podeljene na različite stranice, ali se prelazak sa stranice na stranicu obavlja isključivo korišćenjem JavaScript logike i nikada ne podrazumeva osvežavanje kompletnog prikaza unutar web pregledača.

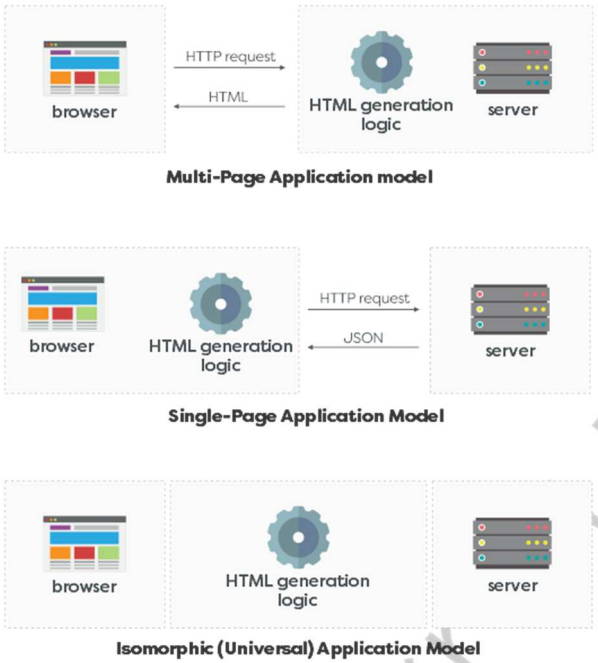
Izomorfne (univerzalne) web aplikacije

Serverska logika web aplikacija se tradicionalno realizuje korišćenjem popularnih backend jezika: PHP, C#, Java, Python... Ipak, sa pojavom izvršnog okruženja Node.js, razvoj serverske logike web aplikacija postao je moguć i upotrebom JavaScript jezika. Tako su stvorene mogućnosti da jedna web aplikacija bude kreirana korišćenjem identičnog jezika i na klijentu, ali i na serveru.

Aplikacije čija se serverska i klijentska logika kreiraju korišćenjem JavaScript jezika nazivaju se izomorfne ili univerzalne aplikacije (engl. *Isomorphic or Universal applications*).

Izomorfna osobina JavaScript jezika često se koristi za nadogradnju Single-page modela. Naime, u prethodnim redovima ste mogli da pročitate da se kod Single-page aplikacija kompletno generisanje prezentacije (HTML koda) obavlja na klijentu. Zbog toga se na inicijalni prikaz sadržaja unutar web pregledača čeka duže nego kod aplikacija koje koriste tradicionalni model, odnosno kod aplikacija koje od servera dobijaju već pripremljen HTML kod. S obzirom na to da Single-page aplikacije od servera dobijaju samo podatke, korisnik mora da sačeka neko vreme, dok JavaScript programska logika na klijentu ne obavi generisanje prezentacije na osnovu takvih podataka. Korišćenje identičnog jezika na klijentu i na serveru omogućava da se ovakav, osnovni nedostatak Single-page aplikacija prevaziđe. Naime, izomorfne aplikacije omogućavaju da se logika za generisanje prezentacije podeli između servera i klijenta. Na serveru se obavlja generisanje najznačajnijih delova, koje je potrebno što pre prikazati korisniku. Generisanje ostalih delova, nakon inicijalnog prikaza, obavlja JavaScript kod koji se nalazi na klijentu.

Različiti načini za generisanje prezentacije, u zavisnosti od modela koji se koristi, ilustrovani su slikom 1.3.



Slika 1.3. Generisanje prezentacije kod različitih tipova web aplikacija

Na slici 1.3. možete da vidite osnovne razlike između različitih modela razvoja web aplikacija, kada je u pitanju generisanje prezentacije. Kod višestraničnih aplikacija, prezentacija se kreira na serveru, dok jednostranične aplikacije podrazumevaju generisanje prezentacije na klijentu. Izomorfne aplikacije omogućavaju da se obavi kombinacija dva prethodno spomenuta modela, i to korišćenjem identičnog jezika i na klijentu, ali i na serveru.

S obzirom na to da izomorfne JavaScript aplikacije podrazumevaju korišćenje JavaScript jezika i na serveru, logika za generisanje inicijalnog prikaza može se obaviti na serveru, što na kraju omogućava klijentu da kao odgovor na inicijalni HTTP zahtev dobije potpuno pripremljen početni prikaz. Ovakva osnovna osobina izomorfnih JavaScript aplikacija za sobom povlači brojne prednosti:

- **lakše održavanje** – s obzirom na to da se identičan jezik koristi i na serveru i na klijentu, logika aplikacije se može efikasno podeliti i na taj način preduprediti ponavljanje identične logike više puta, što na kraju olakšava održavanje,
- **bolje performanse** – korišćenje JavaScripta na severu omogućava da se generisanje HTML koda obavi na serveru, pa da klijent zajedno sa inicijalnim HTTP odgovorom dobije i kompletnu pripremljenu prezentaciju, što ga oslobađa potrebe da takvo generisanje obavlja samostalno; to na kraju podiže performanse, jer korisnik ranije može da dobije inicijalni prikaz,
- **bolji SEO** – generisanje inicijalnih stranica na serveru donosi prednost i prilikom optimizacije stranica za pretraživače (engl. *SEO*); s obzirom na to da se stranice isporučuju od servera, web crawleri mogu lakše da ih indeksiraju,
- **bolji UX** – ukoliko se obavi vešt balans raspodele logike koja se nalazi na serveru i one na klijentu, izomorfne JavaScript aplikacije mogu da poboljšaju i opšti korisnički doživljaj prilikom korišćenja aplikacija; mogućnost pripreme najvažnijih prezentacionih delova aplikacije na serveru znači da će oni brže biti isporučeni korisniku, što povoljno utiče na opšti korisnički utisak o aplikaciji.

Progresivne web aplikacije

Za kraj priče o različitim tipovima današnjih web aplikacija biće definisan još jedan pojam koji se u poslednje vreme često može čuti. Reč je o progresivnim web aplikacijama (engl. *Progressive web applications, skraćeno* ***PWA***).

Progresivne web aplikacije su relativno nov pojam koji je iskovala kompanija Google kako bi definisala web aplikacije koje kombinuju najbolje osobine web i native aplikacija. Stoga, kako bi se neka aplikacija nazvala progresivnom, neophodno je da zadovolji sledeće kriterijume, odnosno da poseduje sledeće osobine:

- **progresivnost** – osobina po kojoj su progresivne aplikacije i dobile naziv, a odnosi se na obavezu takvih aplikacija da bez problema funkcionišu na bilo kom tipu uređaja, odnosno i na jednostavnim uređajima, sa skromnim skupom funkcionalnosti, ali i na uređajima sa naprednijim osobinama i bogatijim skupom funkcionalnosti; pojam progresivnosti odnosi se na mogućnost progresivnog adaptiranja aplikacije na dostupne mogućnosti uređaja,
- **mogućnost lakog otkrivanja i linkovanja** – odnosi se na mogućnost otkrivanja progresivnih aplikacija korišćenjem sajtova za pretraživanje weba; s obzirom na to da su progresivne aplikacije zapravo web aplikacije, one koriste mogućnost weba i HTTP protokola, kako bi korišćenjem URL adresa obezbedile laku dostupnost i deljenje; ovo je jedna od odlučujućih prednosti web aplikacija u odnosu na *native* aplikacije,
- **prilagodljivost** – osobina koja se odnosi na mogućnost prilagođavanja aplikacija bilo kojoj formi, što se prevashodno odnosi na displeje različitih osobina, od displeja na pametnim časovnicima, pa sve do onih na desktop uređajima,
- **mogućnost rada u nepovezanom (engl. *offline*) režimu** – progresivne aplikacije moraju omogućiti nesmetan rad čak i onda kada nema mrežne konekcije,
- **mogućnost instalacije** – moderni web pregledači omogućavaju da se web aplikacija koja ispunjava određene kriterijume instalira na korisničkom uređaju; instalacija se obavlja unutar web pregledača, što na kraju donosi mogućnost pokretanja takve aplikacije na identičan način na koji se pokreću *native* aplikacije (na primer, klikom na ikonicu na desktopu); to ubrzava otvaranje aplikacije i oslobađa od potrebe da se otvara web pregledač i da se unutar njega kuca URL adresa na kojoj je smeštena aplikacija,
- **mogućnost automatskog ažuriranja podataka i novih funkcionalnosti** – progresivne aplikacije moraju uvek biti ažurne, kako po pitanju podataka kojima rukuju tako i kada se govori o njihovim funkcionalnostima, a sve to bez potrebe da korisnik preduzima neke posebne akcije; s obzirom na to da je reč o web aplikacijama, takvo nešto se postiže mnogo lakše nego kod tradicionalnih, *native* aplikacija,
- **sigurnost** – progresivne web aplikacije je neophodno isporučivati preko HTTPS protokola; to je jedan od osnovnih uslova kako bi većina modernih web pregledača omogućila njihovo instaliranje na korisničkom uređaju.

Kreiranje progresivnih web aplikacija primarno podrazumeva korišćenje sledeće tri tehnologije:

- HTTPS – sigurna varijanta HTTP protokola,
- Service Worker API – Web API sa skupom funkcionalnosti koje između ostalog omogućavaju kreiranje aplikacija koje nesmetano funkcionišu u nepovezanom okruženju,
- Web app manifests – fajl koji korišćenjem JSON formata opisuje progresivnu web aplikaciju.

Neki od primera popularnih progresivnih aplikacija su AliExpress, Starbucks, Uber, Pinterest, Spotify, Trivago, Tinder…

Napomena

Bitno je razumeti da upravo opisani tipovi web aplikacija nisu isključivi. To praktično znači da se u praksi web aplikacije često kreiraju kombinovanjem različitih osobina upravo opisanih tipova aplikacija. Stoga prethodne redove ne treba shvatiti kao striktnu podelu, već samo kao smernice koje opisuju najznačajnije osobine danas popularnih modela razvoja koji se koriste u web programiranju.

Rezime

- Web sajtovi sa osobinama kompjuterskih programa drugačije se nazivaju web aplikacije.
- Web aplikacije podrazumevaju dva različita nivoa izvršavanja programskog koda, odnosno izvršavanje na klijentu i na serveru.
- JavaScript aplikacije su web aplikacije za čije kreiranje se u većoj ili manjoj meri koristi JavaScript jezik.
- JavaScript se može koristiti za realizaciju brojnih aspekata web aplikacija, kako na klijentu tako i na serveru.
- JavaScript aplikacije je moguće kreirati kombinacijom JavaScript jezika i Web API-ja, ili upotrebom različitih biblioteka ili softverskih okvira namenjenih frontend programiranju.
- Višestranične aplikacije (MPA) funkcionišu po tradicionalnom modelu, koji podrazumeva da svaki klijentski zahtev rezultira isporukom kompletne HTML stranice.
- Jednostanične aplikacije (SPA) od servera dobijaju samo podatke na osnovu kojih samostalno obavljaju generisanje prezentacije; zbog takve osobine, Single-page aplikacije putanjama rukuju isključivo na klijentu.
- Aplikacije čija se serverska i klijentska logika kreiraju korišćenjem JavaScript jezika nazivaju se izomorfne ili univerzalne aplikacije.
- Progresivne web aplikacije su relativno nov pojam, koji je iskovala sama kompanija Google kako bi definisala web aplikacije koje kombinuju najbolje osobine web i native aplikacija.