Skriptni programski jezici

Tvrdinić, Karlo

Sveučilište Jurja Dobrile u Puli

Odjel za informacijsko-komunikacijske tehnologije

Karlo Tvrdinić

SKRIPTNI PROGRAMSKI JEZICI

Završni rad

Pula, srpanj 2017.

Sveučilište Jurja Dobrile u Puli

Odjel za informacijsko-komunikacijske tehnologije

Karlo Tvrdinić

SKRIPTNI PROGRAMSKI JEZICI

Završni rad

JMBAG: 0303054334, redoviti student

Studijski smjer: Sveučilišni preddiplomski studij Informatika

Predmet: Programiranje

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc.dr.sc. Tihomir Orehovački

Pula, srpanj 2017.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Karlo Tvrdinić, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, 18.09.2017. godine

IZJAVA

o korištenju autorskog djela

Ja, Karlo Tvrdinić dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom Skriptni programski jezici koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 18.09.2017.

Potpis

_____

Skriptni programski jezici

Karlo Tvrdinić

Sažetak: U svrhu završnog rada, implementirani su primjeri u različitim skriptnim programskim jezicima gdje je ilustrirana njihova moć. Također navedene su pojedinosti o konceptu skriptnog programskog jezika te njegovoj svrsi. Osim primjera, odabrani skriptni jezici su objašnjeni i uspoređeni s nekoliko različitih kriterija.

Ključne riječi: skriptni jezici, JavaScript, PHP, Python, Perl.

Scripting programming languages

Abstract: For the purpose of this bachelor's thesis, an example was implemented in different scripting programming languages thus illustrating their power. Details on the concept of scripting programming language and its purpose are provided as well. Apart from the example, the selected scripting languages have been explained and compared by several different criteria.

Keywords: scripting languages, JavaScript, PHP, Python, Perl.

Sadržaj

Popis slika

1. UVOD

Tijekom proteklih nekoliko godina skriptni programski jezici su uvelike napredovali. Desetke godina unazad je skriptni jezik bio promatran kao pomoćni alat koji nije bio pogodan za opće programiranje. Danas je teško izbjeći korištenje skriptnog jezika jer je sadržan gotovo u svakoj web stranici te je također dio većine softverskih sustava. Korištenje skriptnog jezika lakše je nego korištenje programskog jezika, jer omogućava lakši uvid u grešku. Da bi se razvio jednostavan program pomoću skriptnog jezika, potrebno je svega nekoliko sati.

Zbog neinformiranosti korisnici danas sve više vjeruju kako je svaki programski jezik skriptni jezik pa čak i *C*++, tj. vjeruju da se skriptni jezici dijele na dvije vrste: skriptni jezici s kompajlerom i skriptni jezici s tumačem (eng. *interpreter*). Zbog činjenice da je Internet postao dostupan velikom broju ljudi nije neobično da se programiranjem bave i korisnici kojima to nije primarno zanimanje. Većina ih se odluči koristiti skriptni jezik jer ga je lakše i brže naučiti u usporedbi s programskim jezikom.

Skriptni jezici predstavljaju stil programiranja koji je različit od uobičajenog programiranja u programskim jezicima. Njihova je namjena komponiranje programa od gotovih aplikacija čime se postiže viša razina programiranja i brži razvoj aplikacija. Općenito, skriptni jezik se može naučiti brže od strukturiranog jezika kao što je *C* ili *C*++. Jedna od razlika skriptnog jezika i programskog jezika je što programski jezik kompilira cijeli program u izvršnu datoteku prije pokretanja, dok skriptni jezik koristi strojni kôd[1] kojeg učitava i izvršava u trenutku, odnosno pretvara napisani kôd liniju po liniju u strojni kôd koji se izvršava.

Skriptni jezik poput *Perl* i *Tcl*-a predstavlja vrlo različite stilove programiranja od programiranja u jeziku kao što je *C* ili *Java*. Skriptni jezik je dizajniran tako da se može lijepiti s drugim skriptnim programskim jezikom s kojim zajedno tvori jednu cjelinu ili aplikaciju. On može koristiti nekoliko vrsta platformi za postizanje višeg stupnja programiranja. Razvoj aplikacije u skriptnom jeziku obično je brži od razvoja u programskom jeziku.

[1] Strojni jezik – sadrži dva elementa, to su 0 i 1. Računalu razumljiv jezik.

Jedna od velikih prednosti skriptnog jezika u odnosu na programski jezik je nizak stupanj tipiziranja (eng. *weak typing*), dok je programski jezik visoki stupanj tipizacije (eng. *strong typing*). To ga čini mnogo lakšim za korištenje. Programerima se teško nositi s visokim stupnjem tipizacije zbog toga što program može sadržavati tisuće varijabla, a kod visokog stupnja tipizacije svaka varijabla mora biti unaprijed deklarirana tipom varijable kako bi program zauzeo određenu memoriju i kako bi podaci koje koristi bili prepoznatljivi. Snaga niskog stupanja tipiziranja leži u tome što se varijabla ne mora unaprijed deklarirati odnosno navoditi tip varijable, što zasigurno smanjuje nastale greške, te samim time programer štedi na vremenu.

U ovom završnom radu biti će definirano što su skriptni programski jezici te će se opisati njihove mogućnosti. Zatim će na primjeru njegove primjene biti ilustrirana dva odabrana skriptna jezika nakon čega će se predstaviti i dodatno opisati neki najpopularniji skriptni jezici. Nastojat će se ponuditi odgovor na pitanje zašto je skriptni jezik toliko važan u programskom svijetu te koje su sve njegove mogućnosti. Isto tako, pod poglavljem 8 su uspoređeni skriptni jezici po određenim kriterijima.

2. PARADIGME SKRIPTNIH JEZIKA

Kako bi se lakše predočilo što su skriptni programski jezici koji su obrađeni u završnom radu, bit će navedene paradigme četiriju skriptnih jezika: *JavaScript*, *PHP*, *Python* i *Perl*. Neke paradigme se uglavnom odnose na implikacije za izvršni model jezika, kao što je slijed operacija definiran izvršnim modelom.

JavaScript sadrži više paradigmi i objektno je orijentiran više od većine jezika jer se unutar *JavaScript*-a nalaze objekti. *JavaScript* djeluje funkcionalno, funkcije su objekti koje se koriste u kôdiranju. *JavaScript* je imperativ sa sličnom sintaksom kao što ima i *Java*. Dobro je imati na umu da dio *JavaScript*-a ovisi o tome gdje se izvršava.

PHP je dinamičan jezik koji podržava razne programske tehnike. Brzo se razvio i postao popularan, posebice kada su u *PHP* verziji 5 nastale anonimne funkcije, imenski prostori i objektno orijentirani model. *PHP* podržava skup objektno orijentiranih značajki uključujući podršku za apstraktne klase, sučelja, nasljeđivanje, konstruktore, iznimke i još mnogo toga. Podržava funkcionalno programiranje odnosno prvoklasne funkcije što znači da se funkcija može dodijeliti varijabli (Bulger et al., 2014).

Python sadrži razne paradigme te u potpunosti podržava objektno orijentirano programiranje i strukturno programiranje. Mnoge značajke podržavaju funkcionalno programiranje, te aspektno orijentirano programiranje. Mnoge druge paradigme su podržane putem proširenja.

Perl je vrlo koristan jezik, ne zato što je fleksibilan, nego zato što je jezik s više paradigmi, a to znači da se kôd može pisati na razne načine. *Perl* je imperativni, funkcionalni i objektno orijentirani jezik.

Zaključeno je da se skriptni programski jezici sastoje od raznih paradigmi, što je dobro za programera. Na primjeru koji se nalazi u poglavlju 9 uočljiva je primjena objektno orijentiranog načina programiranja kao i funkcionalnog programiranja.

3. SKRIPTNI PROGRAMSKI JEZICI

Tijekom posljednjih nekoliko godina zanimanje programera za skriptne programske jezike se uvelike povećao. U skoroj budućnosti postoji velika vjerojatnost da skriptni programski jezik postane jezgra većine programskih projekata, zbog toga što ima moć lijepljenja već postojećih komponenti koje čine jedan aplikacijski sustav.

Skriptni programski jezici uvelike su napredovali u usporedbi sa stanjem otprije desetak godina kad su ih programeri smatrali pomoćnim alatom zbog toga što sami po sebi nisu bili pogodni za programiranje. Danas je zainteresiranost za skriptni jezik ogromna kako u akademskim krugovima tako i u softverskoj industriji (Kanavin, 2002).

3.1. Što je skriptni jezik ?

Razlike između skriptnog i tradicionalnog programskog jezika su poprilično nejasne. No, moguće je istaknuti obilježja skriptnog jezika koje mogu poslužiti kao definicija, a to su:

- Skriptni jezik se interpretira ili interpretira instrukcije, te se ne kompilira u izvorni kôd
- Upravljanje memorijom brine sakupljač smeća (eng. *garbage collector*), programer ne treba brinuti o memoriji
- Uključuje vrstu podataka visoke razine, kao što su liste, asocijativna polja, itd.
- Okvir izvršenja skriptnog programskog jezika može biti integriran u program koji je već napisan
- Skriptni program može pristupiti modulima napisanim u nižem programskom jeziku, jedan od njih je *C*

Nisu svi skriptni jezici jednaki, oni se razlikuju po mnogočemu. Glavna ideja skriptnog jezika je njegova dinamična priroda koja omogućuje obradu podataka slično kao program (Kanavin, 2002).

3.1.1. Zašto je skriptni jezik važan ?

Najzastupljenija dva aplikacijska programska jezika unazad deset godina bili su *C* i *C++*. No, kako je vrijeme prolazilo skriptni jezici su postali napredniji i bolji od programskih jezika skoro u svim područjima. Većina ostalih zadataka kao što je aplikacijski prototip, te spajanje različitih aplikacijskih modula najbolje odrađuje skriptni jezik. Glavni problem kod *C* i *C++* programskog jezika je to što ne dopušta obradu memorije izvršnom okolinom već zahtijeva njenu ručnu obradu. Potrebno je deklarirati varijable, ručno upravljati pokazivačem po listi i dimenziji među-spremnika (eng. *buffer*), provjeravati ili spriječiti prekoračenje među-spremnika, te je nužno obratiti pozornost na alokaciju i dealokaciju dinamičke pohrane. Često je navedeni problem uzrok velike količine komplikacija, a samim time i pogrešaka. Prekoračenje među-spremnika je česti razlog rušenja programa i sigurnosnih rupa. Neke greške je posebno teško pronaći, kao npr. propuštanje memorije.

Skriptni jezici izbjegavaju upravljanje ručno po memoriji, a to uspijevaju zahvaljujući memorijskom upravljačkom sustavu u dijelu za izvršavanje (eng. *runtime*), tj. programskom tumaču. U nastavku će biti prikazane detaljnije razlike između interpretera i kompajlera.

Više programa može dijeliti jedan interpreter, čime se smanjuje potrošnja memorije. U današnje vrijeme je korisno i smisleno pisati programe na nekoliko jezika, nakon čega se može odabrati najbolji jezik za određene pod-zadatke. Na primjer, dijelovi koji ovise o vremenu i vremenski su kritični, mogu se napisati u *C*-u, pristup podacima se može odraditi preko *SQL*-a, te se sve zajedno spoji u jednu cjelinu. Potom korisničko sučelje koje je na vrhu i prikazuje se korisniku, može se napisati u *Tcl*-u ili *Python*-u (Kanavin, 2002).

3.1.2. Budućnost skriptnog jezika

Kroz desetljeće napretka, postoji velik pad uporabe statičkog jezika[2], kao što je *C++* i *Java*, a raste upotreba dinamičkog jezika kao što je *Python*, *Ruby*, *Perl*, *PHP* i *JavaScript*. Navedeni skriptni jezici će postati neizostavni programski alati za uporabu narednih godina. Primarni razlog za to je da skriptni jezik ne zahtijeva vrijeme za kompiliranje jer ga ne koristi (dok *C++* se može kompilirati ponekad duže vrijeme na izuzetno velikim sustavima). Stoga, čini se da programer treba pridodati više pažnje skriptnim jezicima kao što su *Python* i *Ruby* (Kanavin, 2002).

3.2. Primjena skriptnog jezika

Skriptni jezici kao što su *Perl*, *Python*, *Rexx*, *Tcl* kod *Unix*-a predstavljaju različite načine programiranja od programiranja u sistemskom programskom jeziku kao što je asembler (eng. *assembly*). Primjenom skriptnog jezika treba polaziti od pretpostavke da na repozitoriju već postoji kolekcija od korisnih komponenti koje su napisane u drugom skriptnom jeziku. Skriptni jezik nije namijenjen za pisanje programa od samog početka, on je namijenjen spajanju komponenti kako bi one činile jednu cjelinu, jer se već mnogo gotovih dijelova nalazi na internetu. Na primjer, *Visual Basic* može se koristiti za osjetljivost ekrana kojeg će korisnik koristiti na ekranu, dok *shell* može odrađivati filter same aplikacije. Skriptni jezici rijetko se koriste za složene algoritme i strukture podataka, većinom su korisni pri proširivanju komponenti u aplikaciji. Na osnovu navedenih tvrdnji može se reći da se skriptni programski jezik može nazvati *ljepilo komponenti* ili *jezik integracije sustava* (Ousterhout, 1998).

Kako bi povezivanje komponenti koje se spajaju u jednu cjelinu bilo lakše, skriptni jezici pokušavaju biti platformski odnosno sve komponente izgledaju kao da su međusobno zamjenjive. Kôd i podaci su često i lako zamjenjivi, npr. kôd koji je napisan u *PHP*-u se može lako prebaciti u drugi skriptni jezik. Skriptni jezik je često orijentiran na niz znakova (eng. *string*), jer su jedinstveni za mnogo različitih stvari.

[2] Statički jezik – zahtjeva deklariranje i navođenje tipa varijable prije upotrebe.

Moglo bi se činiti da u radu skriptnog jezika s niskim stupnjem tipiziranja greška može proći neopaženo, no to nije istina. U praksi, skriptni jezik jednako je siguran kao i programski jezik. Razlika je u tome što skriptni jezik obavlja provjeru pogrešaka u krajnjem mogućem trenutku, tj. za vrijeme korištenja vrijednosti koja izazove pogrešku. Pri kompajleru je omogućeno detektiranje pogrešaka pri samom kompiliranju čime se izbjegava trošenje vremena koje bi se izgubilo na prevođenju programa. O razlikama između kompajlera i interpretera će više riječi biti u nastavku (Ousterhout, 1998).

U svakom slučaju programiranje u programskom jeziku zahtijeva mnogo više uloženog vremena i truda te više kôdova za razliku od skriptnog programskog jezika. No, skriptni jezici pružaju manje koristi kada se primjenjuje za programe koji se tek počinju raditi, tj. za prvu implementaciju. Također, prednost skriptnog jezika ovisi o tipu aplikacije koja se izrađuje.

3.3. Različiti alati za različite zadatke

Skriptni programski jezik se teško može zamijeniti s programskim jezikom jer je svatko od njih namijenjen za drugačiju grupu zadataka. Aplikacija koja koristi više komponenti, te lijepljenje dijelova, puno efikasnije i brže se razvija uz skriptni jezik. Kada bi se koristio programski jezik, sustav bi zahtijevao veliku količinu kôda i kôdova konverzije za povezivanje komponenti što nije efikasno, dok za kompleksnije algoritme i strukture podataka bi bilo efikasnije koristiti programski jezik. Kada je brzina ključna, programski jezik može do deset puta brže pokrenuti aplikaciju od skriptnog jezika, jer se troši manje vremena na provjere (Ousterhout, 1998).

Velike računalne platforme su prije trideset godina omogućile programiranje putem skriptnih i programskih jezika. Osamdesetih godina kod *Unix*-a, *C* se koristio za programski jezik, dok je *shell* jezik za skriptiranje. Nakon jednog desetljeća, svijet računala se razvijao i stvorio mnogo inovacija, a s njime su se razvijali programski i skriptni jezici. Devedesetih godina, uz *C* se pojavio i *C++* koji se također koristio za programiranje u programskom jeziku dok je za skriptni jezik razvijen *Visual Basic*. Danas ima puno više izbora pri odabiru jezika za razvoj aplikacije, ovisno o aplikaciji može se odabrati jezik koji više odgovara programeru. Početkom 1991. godine pojavio

se još jedan programski jezik pod nazivom *Java*, a kod skriptnih jezika to su *JavaScript*, *Perl* i *Tcl*. Više o *JavaScript*-u može se pročitati u poglavlju 4. Kombiniranje skriptnog programskog jezika i sistemskog programiranja može tvoriti izuzetno veliku moć. Pomoću sistemskog programiranja može se kreirati zanimljive komponente napravljene u asembleru uz korištenje skriptnog jezika. Na primjer, komponente *ActiveX*[3] mogu se kreirati u programskom jeziku *C*, te se kasnije mogu koristiti u programima pomoću *Visual Basic* skriptnog jezika (Ousterhout, 1998).

3.4. Skriptiranje u usponu

Skriptni jezici postoje već duže vrijeme, no jedno od čestih pitanja je što ih čini tako važnim. Njihovu vrijednost drži nekoliko ključnih faktora. Najvažniji faktor je spajanje komponenti više skriptnih jezika u jednu cjelinu. Objasnit će se primjeri grafičkog sučelja, interneta i komponenti razvojnog okruženja (eng. *framework*).

Prvo će se objasniti primjena grafičkog sučelja (eng. *graphical user interface*). Danas grafičko sučelje koriste gotovo sve aplikacije, te je teško zamisliti aplikaciju bez grafičkog sučelja. Grafičko sučelje je počelo biti popularno početkom osamdesetih godina, naglo se proširilo jer je zainteresiranost brzo rasla. Grafičko sučelje u osnovi je spajanje aplikacija, glavni cilj mu je spojiti grafičku kontrolu i internu funkcionalnost aplikacije. Kada je u pitanju dizajn, bolje i lakše je koristiti skriptni jezik nego programski jezik. Alati koje koristi programski jezik *C* i *C++* nisu lagani za naučiti, nespretni su za uporabu, te nisu fleksibilni u rezultatima koje proizvode. Do komplikacija dolazi kada dizajner treba pisati kôd, tj. kada treba odrediti ponašanje nekog objekta. Za kreiranje grafičkog sučelja najbolje je koristiti skriptni jezik jer je jednostavniji i dozvoljava kombiniranje jezika. Skriptni jezici su sve zastupljeniji pri primjeni grafičkog sučelja, njihova popularnost i važnost raste sve više (Ousterhout, 1998).

Internet je također alat koji spaja dijelove, slično kao i skriptni jezik. On čini veliki broj podataka i stranica dostupnim. Idealni alat za internet je program koji spaja komponente u jednu cjelinu i kombinira ih, te komponente zajedno da funkcioniraju, a

[3] ActiveX – komponenta koja daje funkcionalnost brojnim softverskim aplikacijama

to upravo čini skriptni jezik. Iz tog razloga razvojem interneta također se povećala potražnja za skriptnim jezicima, te se skriptni jezici razvijaju zajedno s Internetom.

Komponente razvojnog okruženja kao što je *ActiveX*, *OpenDoc* je treći primjer zašto su skriptni jezici u porastu. Ako se za manipuliranje komponenata ne koristi skriptni jezik, većina snage komponentnog okvira će se izgubiti jer za spajanje komponenti najbolje je koristiti skriptni jezik koji je i namijenjen za takve poslove. To može objasniti zašto su komponente razvojnog okruženja uspješnije kod primjene na računalima (Ousterhout, 1998).

Dok se je razvijala hardverska komponenta računala, skriptni jezici su također imali koristi od toga, a samim time komponente skriptnih jezika su bile brže. U nekim slučajevima niti programski jezik nije bio dovoljno učinkovit, pa su programeri aplikaciju pisali u asembleru. Osamdesetih godina strojevi su bili mnogo sporiji, no svakih osamnaest mjeseci njihove se performanse udvostruče. Danas mnoge aplikacije mogu biti implementirane interpretiranim jezikom, a da imaju odlične performanse. Primjer, *Tcl* skripta može manipulirati s nekoliko tisuća objekata i dalje može pružati dobar interaktivni odgovor. Što su računala brža, te performanse hardverskih komponenata bolje, skriptni jezici postaju sve brži i učinkovitiji za sve veće aplikacije (Ousterhout, 1998).

Od kada postoje skriptni jezici postoji sve više neformalnih programera. Takvim ljudima programiranje nije glavni posao, obično takvi programeri koriste jednostavne upite za baze podataka, te se koriste jednostavnim proračunskim formulama. Neformalni programeri ili povremeni programeri neće trošiti mjesece za učenje određenog programskog jezika, veća vjerojatnost je da će učiti skriptni jezik zbog njegove jednostavnosti. Obično će programer potrošiti nekoliko sati za učenje skriptnog jezika, te će s postignutim znanjem moći napraviti jednostavan program koji se može koristiti. Programski jezik je teži za naučiti od skriptnog programskog jezika, skriptni jezik ima jednostavniju sintaksu kojom se služi, te izbjegava složene značajke. To je još jedan od razloga zašto su skriptni jezici u usponu.

Danas postoji mnogo programa napisanih u skriptnom jeziku. *Unix* operacijski sustav ima puno više *shell* skripti nego *C* programa, također *Windows* ima više programera i aplikacija u *Visual Basic*-u nego u *C* ili *C++*. Naravno, za veliki program

je korisnije upotrijebiti programski jezik. No, skriptni jezici predstavljaju snagu koja pokreće razvoj aplikacija i efikasniji su od programskih jezika, te se njihovo tržište sve više širi (Ousterhout, 1998).

3.5. Razlika interpretera i kompajlera

U ovom poglavlju će se prikazati razlike između interpretera i kompajlera. Za početak je dobro spomenuti da skriptni jezik koristi interpreter dok programski jezik koristi kompajler, no ne nužno.

Jedno od često postavljenih pitanja unutar informatičke tehnologije je „Koja je razlika između kompajlera i interpretera?“. Naime, računala razumiju samo binarni jezik, tj. jezik koji može samo sadržavati nule i jedinice odnosno vrijednosti istina i laž. Komunikacija s računalom je dakako neusporediva s ljudskom komunikacijom. Da bi se komuniciralo s računalom, te da bi računalo razumjelo jezik kojim ljudi govore potreban je prevoditelj, a prevoditelj se nalazi unutar programskog jezika odnosno skriptnog jezika, te njegova uloga je da prevode naš jezik napisan u nekom programskom ili skriptnom jeziku u strojni jezik. U većini slučajeva kod programskog jezika ulogu prevoditelja čini kompajler, dok kod skriptnog jezika interpreter. Interpreter odnosno kompajler pretvara napisani kôd u binarni ili strojni kôd koje računalo prepoznaje, te računalo izvršava određene instrukcije na osnovu dobivenog rezultata.

Zaključak je da je potreban posrednik kako bi se programski jezik pretvorio u binarni jezik, a tu ulogu ima kompajler. Kompajler je program koji provjerava napisani kôd u programskom jeziku, te prikazuje sve greške koje se nalaze u napisanom kôdu. Kada se ručno poprave ili izbrišu pronađene greške, tek onda kompajler može pretvoriti program u strojni kôd. Računalo može razumjeti strojni kôd, te na osnovu njega izvršava instrukcije koje su raspisane unutar programskog jezika.

Sada kada je pobliže objašnjeno što je kompajler, lakše će se razumjeti interpreter. Razlikuju se u različitom radu. U natuknicama ispod se navode karakteristike kompajlera, zatim interpretera.

Karakteristike kompajlera:

- Kompajler kao ulaz uzima cijeli napisani kôd u programskom jeziku, te obrađuje cijeli program odjednom
- Kompajler generira srednji kôd, kojeg nazivamo strojni kôd
- Kompajler izvršava *if else* i *switch* naredbe brže od interpretera
- Kompajler zauzima više memorije, jer je cijeli strojni kôd smješten u memoriji odjednom
- Program se ne mora kompilirati stalno. Jednom kada se kompilira, može se pokrenuti u bilo kojem trenutku bez ponovnog kompiliranja
- Popis grešaka programer dobije tek kada se cijeli program provjeri
- Jezik koji koristi kompajler teže je otklanjati greške (eng. debugging)
- Kompajler ne dozvoljava da se program pokrene dok se sve greške ne uklone
- Kompajler je učinkovitiji od interpretera, ali ga je teže za provjeravati i otklanjati greške

Karakteristike interpretera:

- Interpreter ne generira središnji strojni kôd
- Interpreter izvršava sporije if else i switch naredbe
- Interpreter ne generira središnji strojni kôd, zbog čega je mnogo učinkovitiji za memoriju, jer je manje zauzima
- Interpreter prevodi liniju po liniju sve dok ne dođe do kraja
- Interpreter radi do prve greške, čim naiđe na prvu grešku ostatak programa se ne interpretira niti se provjerava
- Otklanjanje grešaka je lagano zato što interpreter ide korak po korak, te kad naiđe na grešku ne interpretira dalje kôd
- Interpreter možda nije efikasniji od kompajlera, ali je zato prikladniji za otklanjanje grešaka

Zaključak je da kompajleri djeluju brže zato što odjednom pretvaraju cijeli kôd u strojni jezik razumljiv računalu. No, mana kompajlera je što cijeli kompilirani kôd, odnosno strojni kôd smješta u memoriju čime se zauzima velika količina memorije. Interpreter je jednostavniji za shvatiti, radi na način sličan čitanju. Kreće od prve linije pa sve do zadnje ili dok ne naiđe na prvu grešku. Pomoću interpretera je lakše pronaći grešku. Mana je što je interpreter spor, svaki put kada se pokreće program interpreter prevodi kôd ispočetka liniju po liniju (Kumar, 2015).

3.6. Klijent - poslužitelj

Web skripte pokreću se na jednom od dva mjesta, a to je klijentska (eng. *client side*) ili poslužiteljska strana (eng. *server side*). Klijent strana odnosi se na web preglednik koji korisnik može vidjeti. Server strana web stranice ujedno je i domaćin (eng. *host*). Većina jezika za kôdiranje web stranica dizajnirana je da se pokreće na strani poslužitelja ili na strani klijenta. Čim korisnik zna koji je jezik i na kojoj se strani izvršava odmah u velikoj mjeri može definirati kako on radi (Codeconquest, 2017).

Mnoge stranice koriste jezike za klijentsku i poslužiteljsku stranu. Neke zadatke mogu odraditi oboje jednako, a neke se stvari mogu odraditi samo na klijentskoj strani dok neke samo na poslužiteljskoj strani.

Skripta na klijentskoj strani je dobra za sve što zahtijeva interakciju s korisnikom, npr. neka jednostavna igra. Skriptiranje na strani poslužitelja je dobro za sve što zahtijeva učitavanje dinamičkih podataka, npr. obavijest o korisniku da je prijavljen na stranici.

Primjer koji je izrađen u svrhu ovog završnog rada može se pronaći u nastavku pod poglavljem 9. U primjeru je korišten *JavaScript* kôd koji se izvršava na klijentskoj strani, te *PHP* koji se izvršava na poslužiteljskoj strani. Uočljivo je kako dva skriptna jezika tvore jednu dinamičnu web stranicu.

3.6.1. Klijentska strana

Pravi primjer skripte koja predstavlja klijentsku stranu (eng. *front end*) je jezik *JavaScript*, o njemu se može proučiti detaljnije u ovom radu pod poglavljem 4. *JavaScript* se dodaje uz *HTML* i *CSS* kôd. Razlog zašto je *JavaScript* na strani klijenta je zato što se njegove skripte izvršavaju kod klijenta nakon što se stranica učita sa servera, no danas ima puno jezika koji se izvršavaju na klijentskoj strani.

Ako se upiše kôd unutar *JavaScript* oznaka „document.getElementById('test').innerHTML = 'Pozdrav';“ onda će se 'Pozdrav' pojaviti u određenom elementu koji je pod *ID*-om 'test'. Sada je sadržaj unutar elementa

pod *ID*-om 'test' zamijenjen. No, ako se otvori izvorni kôd stranice, vidjet će se da je sadržaj koji je bio u elementu sa zadanim *ID*-om promijenjen, to je odradio preglednik kod klijenta, te se skripta izvršila na klijentskom računalu.

3.6.2. Poslužiteljska strana

Unutar poslužiteljske strane (eng. *back end*), skripta se pokreće prije nego što se *HTML* učita na stranici, dakle mora se znati razlika između prije i poslije. Poslije učitavanja *HTML*-a klijent dobiva gotovu stranicu, a prije nego što se *HTML* učita klijent šalje upit koji se šalje serveru da dohvati određenu stranicu.

Danas postoji veliki broj jezika koji se koriste na poslužiteljskoj strani, *PHP* je jedan od najpopularnijih jezika, kao i *Ruby on Rails* i *ASP.NET*. Razlog zašto se zove skripta na poslužiteljskoj strani je taj što se skripte ne pokreću na računalu klijenta, nego se pokreću na serveru odnosno domaćinu stranice, potom se šalje klijentu obrađeni *HTML* kôd koji se izvršio na poslužitelju. S druge strane, *PHP* kôd koji koristi poslužitelj klijent neće moći nigdje vidjeti, dok kôd koji se nalazi na klijentskoj strani moći će vidjeti. To je zato što poslužitelj vodi brigu o *PHP*-u, te rezultat što se šalje korisniku je čisti *HTML* kôd. Više o *PHP*-u može se proučiti u poglavlju 5.

4. JAVASCRIPT

Ovo poglavlje posvećeno je *JavaScript*-u koji je jedan od najpopularnijih skriptnih jezika na klijentskoj strani. *JavaScript* se kombinira sa statičnim *HTML*-om, pomoću kojeg se dodaje interaktivni elementi. *JavaScript* je magični alat kojim se mogu napraviti mnoge stvari ako se koristi ispravno, te ako se koriste određeni aspekti. Drugi naziv za *JavaScript* je *ECMAScript.* Napravio ga je Brendan Eich u *Netscape*-u.

Mnogo ljudi ne proučavaju prvo *JavaScript* nego proučavaju *jQuery*, iznimno je popularan i poznat kao *JavaScript* biblioteka. Razlog tome je što većina ljudi ne želi učiti sintaksu *JavaScript*-a koja je malo zbunjujuća.

Da bi se postalo programer u *JavaScript*-u potrebno je osnovno znanje u *HTML*-u kao i osnovno znanje u programiranju kako bi se kreirala skripta koja ima svoju funkcionalnost, jer *JavaScript* koristi elemente pomoću *HTML* oznaka.

JavaScript ima svojih prednosti i nedostataka. Jedna od prednosti *JavaScript*-a je izvršavanje skripte na strani klijenta što zamjenjuje web server. *JavaScript* je jezik kojeg je relativno lako za naučiti, a sastoji se od sintakse koja je bliska engleskom jeziku. Upotrebljava *DOM* model koji nudi brojne unaprijed napisane funkcionalnosti za različite objekte. *JavaScript* je relativno brzi jezik za krajnje korisnike jer se kôd izvršava na računalu klijenta, rezultat i obrada ovisi o zadanom zadatku. Također, pruža proširenu funkcionalnost za web stranice (Crockford, 2008).

Jedan od glavnih nedostataka *JavaScript*-a je sigurnosni problem. Mali program koji je napravljen preko *JavaScript*-a može biti maliciozan[4], kada se jednom doda maliciozna skripta na web stranicu korisnikov preglednik izvršava JavaScript koji može prouzročiti štetu. Iako postoje određenja ograničenja modernih web standarda za preglednike, zlonamjeran kôd i dalje može biti izvršen u skladu s postavljenim ograničenjima. Drugi nedostatak je što hardverske komponente u računalima nisu kod svih jednake, stoga se *JavaScript* može izvršiti drugačije od računala do računala što može rezultirati nedosljednom funkcionalnošću i sučeljem (Crockford, 2008).

[4] Maliciozan program – softver koji radi štetu korisniku.

4.1. Što je JavaScript ?

JavaScript je danas sadržan u većini preglednika, kao što je *Explorer*, *Chrome*, *Mozilla*, *Opera*, *Safari*, itd. Za *JavaScript* se može reći da je jedan od tri glavna jezika koji se upotrebljava unutar web stranice. Prvi jezik je *HTML* koji kontrolira strukturu web stranice, drugi je *CSS* preko kojeg se radi dizajn web stranice ili teksta koji je napisan unutar *HTML*-a, a treći je *JavaScript* koji je odgovoran za dodatna ponašanja i interakcije na stranici, što čini web stranicu zanimljivijom. Na primjer, kako bi korisnik bio u mogućnosti odabrati sliku te je zumirati na određene načine, potreban je *JavaScript*.

JavaScript je skriptni jezik, koji se dodaje kao skripta unutar web stranice. Kao što je spomenuto skriptni jezik se kombinira s drugim skriptnim jezikom, te kombiniranjem dvaju ili više skriptnih jezika čini jednu cjelinu. To znači da je *JavaScript* limitiran, nema iste mogućnosti kao što posjeduje *C* ili *C*++, da može komunicirati direktno s bazom podataka ili manipulirati datotekama na računalima. Njegova svrha je da web stranicu učini interaktivnom, te da manipulira web stranicama.

Mnogi korisnici misle da su *Java* i *JavaScript* slični jezici, te da je *JavaScript* samo nadogradnja *Jave*, no to nije istina. *Java* i *JavaScript* su različita dva jezika i svaki od njih je namijenjen za različite zadatke.

4.2. JavaScript je klijentski jezik

JavaScript je skripta koja se koristi na strani klijenta, ukratko to znači da se izvršava na klijentskom računalu odnosno na klijentskom pregledniku. Kada se upiše *URL* u klijentski preglednik, on se šalje serveru te se dobivaju informacije od servera koje se šalju natrag klijentskom pregledniku koji se prikazuje u kombinaciji *HTML*-a, *CSS*-a i *JavaScript*-a. Tri navedena jezika tvore jednu cjelinu koja se prikazuje na klijentskom pregledniku.

JavaScript kao skripta na strani klijenta se razlikuje od skripti na strani poslužitelja, kao što je *PHP*, *ASP*. Oni se izvršavaju na serveru upisivanjem *URL*-a na pregledniku.

PHP kôd će se izvršiti na serveru, zatim se rezultat izvršenog *PHP*-a kôda šalje preko servera natrag na računalo odnosno preglednik. Cijelo poglavlje 5 je posvećeno *PHP*-u gdje je detaljnije objašnjen sam skriptni jezik *PHP*.

4.3. Zašto JavaScript ?

JavaScript je važan jezik zato što je jezik web preglednika koji je dio svakodnevice mnogim ljudima, a upravo je on medij koji nam čini podatke interaktivnim i zanimljivim. Njegov način primjene unutar web preglednika čini *JavaScript* jednim od najpopularnijih jezika u svijetu. Mnogi programeri danas se odluče programirati u *JavaScript*-u, jer je lakše naučiti *JavaScript* od *C* ili *C*++, te su programeri sa znanjem *JavaScript-a* danas traženi u većoj mjeri od programera koji imaju znanja u *C* ili *C*++. *JavaScript* je jezik koji nije omiljen među nekim programerima radi povremene prisiljenosti na rad u njemu zbog okruženja koje podržava samo njega. Na primjer, ako programer kvalitetnije radi u nekom drugom jeziku i treba programirati u određenom okruženju, a to okruženje podržava samo *JavaScript*, onda je programer prisiljen koristiti *JavaScript* (Crockford, 2008).

Zanimljiva činjenica kod *JavaScript*-a je što se može napraviti dobar posao s malo znanja o *JavaScript*-u, nije potrebno veliko znanje u području programiranja. To je jezik s ogromnom ekspresivnom moći.

4.4. Prednosti i mane JavaScript-a

JavaScript je izgrađen na osnovu nekoliko dobrih i loših ideja. Dobre su implementacija funkcije, nizak stupanj tipiziranja, te dinamični objekti, a loša ta što je model baziran na globalnim varijablama.

Današnji način upotrebe programskog jezika zahtjeva visoki stupanj tipiziranja. Teorija visokog stupnja tipiziranja dozvoljava pronalazak greški kompajleru da provjeri velike klase za vrijeme kompiliranja. Što se prije uoče greške i isprave, to je bolje za razvoj aplikacije jer će se uštedjeti mnogo vremena. To može biti upozorenje za

programere koji prelaze s programskog jezika kao što je *C++* na *JavaScript*. Naravno, visoki stupanj tipiziranja ne eliminira potrebu za pažljivim testiranjem (Crockford, 2008).

JavaScript može biti povezan kao zlonamjeran zbog odabira ključnih ideja. No, tu je jedan izbor koji je bio posebno loš, a to je da *JavaScript* ovisi o globalnim varijablama za povezivanje. Sve varijable na najvišoj razini svih kompilacijskih jedinica su u jednom imenskom prostoru zvan globalni objekt. Globalne varijable su loše i lako pristupačne, a *JavaScript* je temelj takvih varijabli što znači da je to vrlo loše (Crockford, 2008).

4.5. Primjena JavaScript-a

Ukoliko klijent želi da web stranica bude moćna i kreativna potrebna je primjena *JavaScript*-a. Međutim, *JavaScript* je malo složeniji za postizanje rezultata od *HTML*-a i *CSS*-a. Kao primjer, prikazat će se kako primijeniti *JavaScript*, tako da je kreirana jednostavna skripta unutar *HTML*-a koja ispisuje „Hello World!“.

Da bi se koristio *JavaScript* jezik dovoljno je imati web preglednik i tekst editor. Sve što treba unutar *HTML*-a oznaka (eng. *tag*) je staviti oznake „script“, primjer se nalazi u Slici 1.

```
1  <html>
2      <head>
3      </head>
4      <body>
5          <h1 id="demo"></h1>
6          <p>Primjer skripte</p>
7          <script>
8              // JavaScript se nalazi unutar script tagova
9          </script>
10     </body>
11 </html>
```

Slika 1. Primjer primjene JavaScript

Prvo se kreirala datoteka „index.html“, te je unutar *HTML* datoteke ubačen *JavaScript* koji trenutno ništa ne radi. Sada kada su ubačene „script“ oznake, unutar njih napisan je kôd koji se nalazi na Slici 2.

```
<h1 id="demo"></h1>
<p>Primjer skripte</p>
<script>
    document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

Slika 2. Primjer ispisa pomoću JavaScript

Kada se spremi „index.html“ te osvježi stranica na pregledniku klijenta, prikazat će se „Hello world!“ unutar oznaka „<h1>“.

„Script“ oznake nalaze se pri dnu *HTML*-a, a razlog tome je što se *HTML* izvršava redom kako se pojavljuje kôd u datoteci. U slučaju da se stave *JavaScript* oznake u samom početku, *JavaScript* će se učitati prvi i time će utjecati na *HTML* ispod njega, možda neće funkcionirati, jer će *JavaScript* biti učitan prije *HTML*-a na kojem bi trebao raditi. Stoga, pisanje *JavaScript*-a pri kraju *HTML* stranice je često dobra praksa.

5. PHP

PHP je danas jedan od najpopularnijih skriptnih jezika na strani poslužitelja, te je široko korišten otvoreni kôd (eng. *open source*), namijenjen je za web programiranje, kreiranje dinamičkih web aplikacija, te se može ugraditi unutar *HTML*-a.

PHP je razvijen u *C* programskom okruženju 1994. godine od strane Rasmus Lerdorf-a u svrhu brojanja posjetitelja na svojoj web stranici. U početku *PHP* se zvao *PHP/FI* (*Personal Home Page Tools/Forms Interpreter*) te je tada predstavljao skup *perl* skripti (Gilmore, 2001).

Ono što *PHP* kao skriptni jezik izdvaja od drugih skriptnih jezika je slično kao i *JavaScript* koji je na strani klijenta, a to je jednostavna sintaksa, samo što se kôd izvršava na serveru. Kôd koji je poslan serveru se generira i pretvara u *HTML* oblik koji se potom šalje klijentu. Klijent će primiti rezultat koji se pokrenuo i izvršio na serveru, ali neće znati koji kôd stoji iza njega, jer se kôd koji se izvršava na serveru ne vidi, tako da klijent samo dobiva rezultat.

Jedna od dobrih stvari je što *PHP* nije teško naučiti, te je iznimno jednostavan za nove programere dok za profesionalne programere nudi mnoge napredne značajke, programer s *PHP*-om može učiniti razne stvari.

Navest će se nekoliko prednosti i nedostataka *PHP* skriptnog jezika. Jedna od prednosti *PHP* skriptnog jezika je ta što je *PHP* otvoreni kôd, te ga iz tog razloga velika skupina razvojnih programera razvija što pomaže pri stvaranju brojnim proširenjima kao što su biblioteke. Brzina je isto jedna od prednosti, *PHP* je relativno brz jer koristi mnogo resursa iz sustava. Lagan je za naučiti, koristi sličnu sintaksu kao i *C* programski jezik, a programer koji je upoznat s programskim jezikom *C* lako može napisati web skriptu. Budući da *PHP* održavaju mnogi razvojni programeri, može se reći da je stabilan skriptni jezik. Ima snažnu podršku biblioteka, može se lagano pronaći određeni funkcionalni modul. Pomoću *PHP*-a može se lagano stvoriti konekcija prema bazi, tako da se samim time uvelike smanjuje vrijeme pisanja kôda kod web aplikacija. Isto tako, prednost *PHP*-a je što se izvršava na mnogim platformama, uključujući *Windows*, *Linux* i *Mac* (Bulger et al., 2014).

Jedan od nedostataka *PHP*-a je sličan kao i kod *JavaScript*-a, a to je sigurnost. Budući da je *PHP* jezik otvorenog kôda, programer ima uvid u izvorni kôd. Ako postoji neki nedostatak u izvornom kôdu, programer može taj nedostatak iskoristiti za istraživanje slabosti *PHP*-a. Drugi nedostatak je što *PHP* nije prikladan za velike aplikacije, *PHP je teško* za održavati jer nije modularan. Sljedeći nedostatak je nizak stupanj tipiziranja, implicitna konverzija može iznenaditi neoprezne programere i dovesti do neočekivanih grešaka (Bulger et al., 2014).

5.1. Povijest PHP-a i njegove verzije

PHP se povezuje s *HTML*-om te zajedno čine jednu dinamičku web aplikaciju. Kao svaki skriptni jezik, *PHP*-u je bilo potrebno prisustvovanje *PHP* procesora. *PHP* kôd je pokrenut kao čisti tekst[5] (eng. *plain text*), kao skripta koja se pokreće samo na *PHP* računalima (programski jezik može stvoriti samostalnu binarnu izvršnu datoteku). *PHP* koristi većinu sintakse iz *C*, *Jave* i *Perl*-a. *PHP* se koristi unutar web stranice, stoga može se donijeti zaključiti da je *PHP* skriptni jezik koji se može prikazivati na većini operacijskih sustava i na većini Web servera (Gilmore, 2001).

U početku kada je Lerdorf razvio *PHP*, onda je *PHP* služio za osobnu početnu stranicu (eng. *Personal Home Page*), jer je u početku bio razvijen za potrebe Lerdorf-a da prati posjetitelje na svojoj Web stranici. Lerdorf je kombinirao *PHP* sa svojim vlastitim tumačem (eng. *Form Interpreter*), to je razlog zašto je prva verzija *PHP*-a bila nazvana *PHP/FI* (*PHP 2.0*). Dva programera su 1997. godine svojom obnovom *PHP*-a promijenila akronim *PHP/FI* u *PHP: HyperText Preprocessor*, *Zeev Suraski* i *Andi Gutmans* su tada obnovili PHP jezgru. Prva verzija koja je postala popularna bila je pod nazivom *PHP 3* koja je puštena 1998. godine. *PHP 4* je pušten u svibnju 2000. godine s novom jezgrom, gdje je poboljšana brzina i pouzdanost naspram *PHP 3*. *PHP 4* je poznat također pod imenom *Zend Engine 1.0. PHP 4.* Nove mogućnosti u *PHP 4* su bili tipovi podataka *boolean*, izlazni među-spremnik, nove funkcije s poljima, prošireno objektno orijentirano programiranje, i još mnogo toga (Gilmore, 2001).

[5] Čisti tekst (Plain text) – tekst koji nije računalno označen, posebno oblikovan ili napisan u kôdu

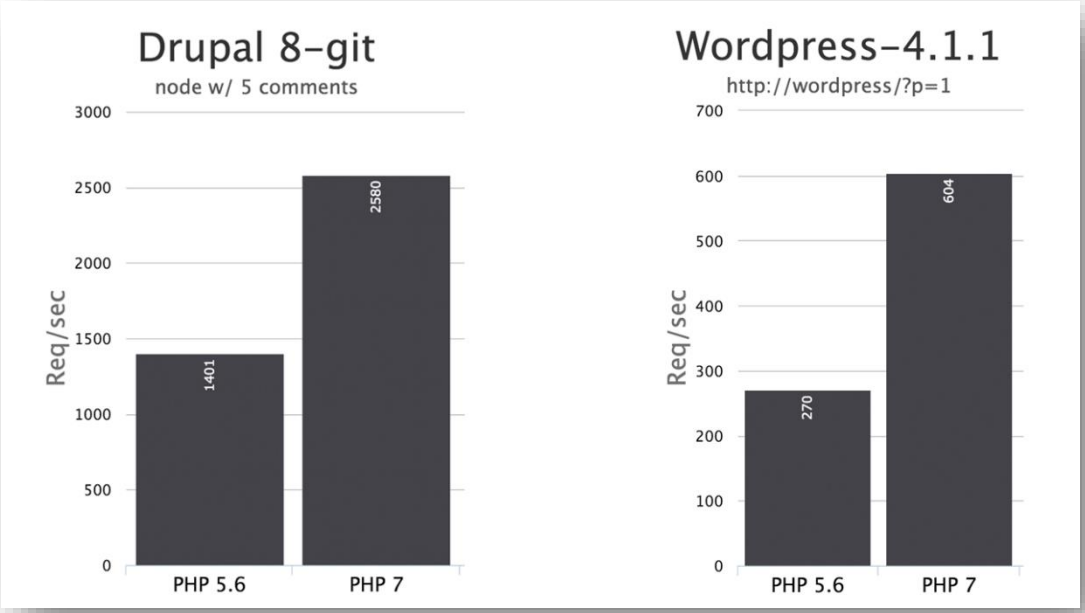
Nova nadogradnja *Zend Engine* je bila 2004. godine, te je nadogradnja poznata pod *PHP 5*. Nove značajke *PHP 5* su:

- Poboljšano objektno orijentirano programiranje
- Ugrađen *SQLite*
- Podrška za nove *MySQL* značajke
- Rukovanje iznimkama pomoću strukture *try-catch*
- Integrirana *SOAP* podrška
- Bolji *XML* alati
- Pokazivači (eng. *iterators*)

PHP 7 je novo izdanje koje ima mnogo poboljšanja naspram *PHP 5* verzije. Zašto *PHP 7*, a ne *PHP 6*? Ukratko, *unikôd*[6] nije bio tako dobar. Kao i kod mnogih projekata, zahtjevi nisu bili dobro definirani što je dovelo do neslaganja, a potom i obustavljanja projekta. Osim *unikôd*-a za kôdiranje, gotovo sve značajke o kojima se raspravljalo za *PHP 6* na kraju je implementirano u *PHP 5.3*. Kada su se nadogradnje prihvatile za veće izdanje, odlučeno je da se izbjegne zbunjenost s projektom koji je zastao unutar *PHP 6* i preskoči odmah na verziju *7* za najnovije izdanje. Jedna od najznačajnijih prednosti u verziji *7* je brzina. Programeri su mnogo radili da promijene *PHP codebase*[7] kako bi smanjili potrošnju memorije i povećali performanse, u čemu su i uspjeli. Istraživanja su pokazala koliko je *PHP 7* brži od *PHP 5.6.* Navedeni rezultati nisu zajamčeni za druge projekt, *PHP* je testiran na velikim projektima, *Drupal* i *WordPress* (Holigan, 2016).

[6] Unikôd – Međunarodno standardno kôdiranje za uporabu s različitim jezikom i skriptom

[7] Codebase - Odnosi se na zbirku izvornog kôda koji se koristi za izgradnju određenog softverskog sustava, aplikacije ili softverske komponente.



Slika 3. Brzina PHP 7 (Holigan, 2016)

Kao što se vidi iz rezultata koji je prikazan na Slici 3, *PHP 7* može primiti dva puta više zahtjeva po sekundi dok je slučaj kod *Wordpress*-a skoro tri puta napredniji *PHP 7* od *PHP 5.6.*

5.2. Primjena PHP-a

PHP je skriptni jezik čija je svrha ranije poznata. *PHP* kao skriptni jezik se kombinira s drugim skriptnim jezikom te time stvara jednu aplikaciju. U ovom poglavlju se objašnjava kako se *PHP* primjenjuje kao skriptni jezik unutar *HTML*-a.

Programe pisane u skriptnom jeziku *PHP* nije potrebno prevoditi u izvršni kôd (eng. *compile*), *PHP* se izvodi unutar interpretera. Koristi se pri izradi web stranica, te se kombinira s *HTML*-om. Kako bi se izradila dinamička web stranica koja će komunicirati sa serverom, a na serveru se nalazi baza podataka, obično se koristi skriptni jezik *PHP*. Unutar primjera koji je priložen u završnom radu *PHP* dohvaća podatke iz baze podataka te se vraća klijentu *HTML* stranica koja sadrži podatke preuzete iz baze podataka. Drugi primjer je vidljiv u poglavlju 9.

PHP se smješta unutar *HTML* oznaka. Oznaka za početak korištenja *PHP*-a je „<?php“, dok oznaka za završavanje *PHP* skripte je „?>“. Sve što je unutar oznaka za početak i kraj *PHP*-a se smatra *PHP* skriptom, te unutar njih se piše *PHP* kôd, ako se isti kôd *PHP*-a napiše izvan *PHP* oznaka, preglednik će kôd gledati kao običan *HTML* tekst. Datoteka u kojoj se nalazi *PHP* mora se naznačiti s ekstenzijom „.php“ kako bi poslužitelj znao da se unutar datoteke nalazi *PHP* kôd. Slika 4 prikazuje jedan mali primjer *PHP* programa koji ispisuje tekst (eng. *string*).

```
<html>
<head>
  <title>PHP program</title>
</head>

<body bgcolor="#ffffff">
  <?php echo "Moj prvi PHP program"; ?>
</body>
</html>
```

Slika 4. Primjer dodavanja PHP skripte

Kao što je vidljivo Slika 4 prikazuje kôd koji je unutar *HTML*-a. *PHP* kôd se izvršava tek kada dođe do poslužitelja, nakon čega se klijentu vraća čisti *HTML*. Slika 5 prikazuje *HTML* kôd koji je poslan klijentskom poslužitelju.

```
<html>
        <head>
          <title>PHP program</title>
        </head>

        <body bgcolor="#ffffff">
          Moj prvi PHP program  </body>
        </html>
```

Slika 5. HTML s izvršenim PHP kôdom

Unutar priložene Slike 5 se vidi kôd koji je generiran na strani poslužitelja, te je kôd koji dolazi do klijenta čisti *HTML*, dok *PHP* kôd korisnicima nije vidljiv.

5.3. Svrha PHP-a

Danas je teško zamisliti web aplikaciju bez korištenja baze podataka. Da bi web stranica koristila bazu podataka jedan od najpopularnijih jezika za komunikaciju odnosno interakciju s bazom je *PHP*.

Ukoliko programer želi napraviti jednu kvalitetnu i profesionalnu web stranicu, vjerojatno će se web stranica bazirati na *PHP*-u i *MySQL*-u. Ako se programer odluči na navedene jezike onda će stranica imati sve potrebne funkcionalnosti za profesionalnu stranicu koja dohvaća određene podatke i prikazuje ih korisniku, mijenja ih, te ih briše iz baze podataka. *PHP* i *MySQL* jednostavni su za koristiti, brzi, besplatni i moćni. Ukoliko se želi napraviti dinamična web stranica u kratkom vremenskom roku *PHP* je najbolji izbor.

Baza podataka se koristi za povezivanje podataka razvrstanih u tablice, a u njoj nema ponavljanja odnosno redundancije podataka[8]. Tablica u sebi sadrži redak, polje i atribut. U jednoj tablici prikupljaju se podaci istog skupa, na primjer podaci o državi, ti podaci se pohranjuju u jednu tablicu. Baza podataka može sadržavati veliki broj podataka do kojih je moguće doći upitima iz *PHP*-a preko kojeg se dohvaćaju podaci na dinamičnu web stranicu. *PHP*-om se mogu podaci unositi, izmjenjivati, te brisati. Baza podataka je opširna tema o kojoj neće biti puno govora u ovom radu (Bulger et al., 2014).

[8] Redundancija podataka – to su podaci koji se ponavljaju, odnosno isti podatak se drži na više mjesta.

6. PYTHON

Često pitanje je zašto ljudi koriste *Python*? Odgovor na pitanje nalazi se u ovom poglavlju. Naime, mnogo programskih jezika je danas dostupno, pa je iz tog razloga postavljeno pitanje zašto baš *Python*, obzirom da trenutno postoji oko 1 milijun korisnika raznih jezika, teško je pronaći točan odgovor na to pitanje.

Po knjizi *Mark Lutz*-a (2013) ispitano je preko 4000 studenata kroz proteklih 16 godina u 260 grupa. Studenti su odgovorili da je jedan od važnih čimbenika kvaliteta softvera. Također su naveli produktivnost programskog okuženja kao važan čimbenik koji *Python* čini tako popularnim. Za razliku od *C*, *C++* i *Jave*, *Python* ima veću produktivnost razvojnog programera. Obično *Python*-ov kôd je manji od *C*, *C++* i *Jave* što znači da je potrebno manje vremena za otklanjanje grešaka, te manje vremena za održavanje. *Python* se može pokrenuti na svim glavnim računalnim platformama. Na primjer, prijenos programskog kôda iz *Linux*-a na *Windows*, obično se samo kopira kôd skripte između strojeva (Lutz, 2013).

Python je usredotočen na čitljivost, koherenciju i kvalitetu softvera, te se općenito razlikuje od drugih alata u svijetu skriptiranja. Za razliku od drugih jezika *Python* ima dizajniran kôd da bude čitljiv i lako razumljiv, a time ponovno upotrebljiv i održiv. *Python* je lako razumljiv čak i ako programer nije autor programa. *Django*[9] je jedan od najpopularnijih razvojnih okruženja za *Python*. Pruža brzo razvijanje aplikacije uz manje kôda naspram drugih programskih jezika. Raspolaže s velikim brojem resursa, pruža ugrađeni okvir testiranja pomoću kojeg programer uklanja greške što mu omogućava brži razvoj aplikacije.

Python je programski jezik opće namjene koji se često primjenjuje u skriptnim ulogama. Obično je definiran kao objektno orijentirani skriptni jezik. Definicija objektno orijentirani skriptni jezik kombinira podršku za objektno orijentirano programiranje (OOP) s cjelokupnom orijentacijom prema skriptnim pravilima. No, *Python* je poznat pod nazivom programski jezik opće namjene koji kombinira proceduralne, funkcionalne i objektno orijentirane paradigme (Lutz, 2013).

[9] Django - razvojno okruženje koje služi za izradu web aplikacije, razvojno okruženje je besplatno i otvorenog kôda

Python ima mnogo prednosti, ali i nedostataka. *Python* je lagan za naučiti, s lakoćom se može izvršiti puno složenih funkcija zahvaljujući standardnoj biblioteci. *Python* se može izvršiti na različitim platformama. Sadrži paradigmu objektno orijentiranog programa, te sadrži mnogo razvojnih okruženja koji ga čine veoma fleksibilnim. Jedan od nedostataka kod *Python*-a je brzina. Mnogo je sporiji od *JavaScript*-a ili *PHP* skriptnog jezika. *Python* je dobro koristiti kod platforma kao što je desktop ili server, ali kod razvoja aplikacije za mobilno računalstvo je manje koristan. Gotovo je nemoguće izraditi *3D* igru visoke grafike pomoću *Python*-a, isto tako ima ograničenja kod pristupa na bazu podataka (Lutz, 2013).

6.1. Povijest Python-a

Razvoj *Python*-a je počeo kasne 1989. godine. Razvio ga je Guido van Rossum u *CWI* centru u Nizozemskoj. Objavljen je za javnu distribuciju početkom 1991. godine. Isto kao *C, C++, Java* i *Perl, Python* je došao iz istraživačke pozadine gdje je programer imao problema s dobivanjem posla s postojećim alatima koji su se tada mogli koristiti (Lutz, 2013).

U to doba, Rossum je bio istraživač znatnim iskustvom u jezičnom dizajnu s interpretiranim jezikom *ABC*, također taj jezik dolazi s *CWI*-a. Bio je nezadovoljan sa sposobnošću jezika kojim je raspolagao pa je htio razviti nešto više. Nakon što je to i učinio, djelomično je razvio jezik višeg nivoa poput *ABC*-a, no vraćanje natrag u *C* nije bila atraktivna mogućnost. Neki od alata koje je Rossum zamislio bili su za obavljanje općih poslova administracije sustava. Imao je želju pristupiti snazi sustavu koji su bili dostupni putem distribuiranog operacijskog sustava zvanom *Amoeba*. Iako je Rossum pomislio na jezik specifičan kao *Amoeba*, generalizirani jezik imao je više smisla, stoga se je krajem 1989. godine počeo razvijati *Python* (Lutz, 2013).

6.2. Što Python može ?

Python je dobro osmišljen programski jezik koji je koristan za zadatke kao u stvarnom svijetu. Obično *Python* se koristi u različitim domenama, kao što je alat za skriptiranje drugih komponenti i implementiranje samostalnih programa. Kako je *Python* jezik opće namjene može se koristiti gotovo za sva područja programiranja, od razvoja web stranica, igara, do kontrole robotike, pa čak i svemirske letjelice.

Međutim, *Python* se svrstava obično u nekoliko širokih kategorija. U nastavku će se objasniti koje su to kategorije u kojima *Python* ima glavnu ulogu.

Prva kategorija za koju se *Python* koristi je programiranje sustava, ugrađuje sučelja za operacijske sustave koje ga čine idealnim za održavanje sustavnih alata i za administriranje sustava. Programi *Python*-a mogu pretraživati datoteke i mape, koje pokreću drugi programi (Lutz, 2013).

Python-ova jednostavnost i brzina čine ga dobrim jezikom za grafičko sučelje koje se koristi za radnu površinu. Dolazi sa standardnim objektno orijentiranim sučeljem zvanim *tkinter*[10] koji mu omogućuje da implementira portabilno grafičko sučelje s modernim izgledom. *Tkinter* grafičko sučelje se pokreće na platformi *Windows, Unix, Linux* i *Mac OS*. Postoje mnogi drugi programi koji se koriste zajedno s *Python*-om za grafičko sučelje. *Python* dolazi sa standardnim internetskim modulima koji omogućuju *Python* programu da obavlja širok raspon mrežnih zadataka, u načinu rada: klijent i poslužitelj jer skripte mogu komunicirati putem slojeva. Ekstrakt informacija koji se šalju serveru u obliku *CGI skripata*[11], slanje datoteka putem *FTP*-a, analiza i generiranje *XML* i *JSON* dokumenata, slanje, primanje, sastavljanje elektroničke pošte. To je samo mali dio mogućnosti *Python* jezika, samim time može se lako donijeti zaključak o *Python* jeziku, te je jasno zašto je tako popularan (Lutz, 2013).

[10] Tkinter – Python-ov standardni paket za grafičko sučelje, objektno orijentirani sloj
[11] CGI skripta – program koji se pokreće na web poslužitelju

6.3. Python kao skriptni jezik

Python je svakako sposoban za skriptiranje, no bolje je *Python* shvatiti kao skriptni jezik u tradicionalnom smislu. On je programski jezik opće namjene koji se često primjenjuje u skriptnim ulogama. Obično se definira kao objektno orijentirani skriptni jezik koji kombinira podršku objektno orijentiranog jezika s cjelokupnom orijentacijom prema ulogama skriptiranja.

Skriptni jezici često koriste interpreter. Razlog tomu je mala koristi od kompiliranja programa, budući da je velika vjerojatnost da će se često mijenjati skripta, program bi se morao svaki put kompilirati i pretvarati u strojni kôd, što bi zahtijevalo dosta vremena. Ako se želi napraviti dobar i profesionalan program s nekim jezikom, nastojat će se izbjeći korištenje interpretera pri velikim kôdovima zbog njegove sporosti. *Python* se smatra skriptnim jezikom zbog povijesnog zamućenja između skriptnog jezika i programskog jezika opće namjene. No, *Python* nije skriptni jezik, već programski jezik opće namjene koji također funkcionira kao skriptni jezik, te iz tog razloga se *Python* nalazi u ovom završnom radu kao skriptni jezik (Lutz, 2013).

6.4. Primjena Python-a

Python se može koristiti za razne svrhe, kao što je izrada aplikacija, web stranica, potpora znanosti, educiranje, te za izradu grafičkog sučelja. Demonstrirat će se mali primjer *Python*-a kod primjene web stranice. Kako bi programer mogao koristiti *Python* za kreiranje web stranice najbolji izbor je koristiti *Django* razvojno okruženje. Postoji mnogo drugih razvojnih okruženja sličnih kao *Django*, ali za izrađivanje web stranice odnosno aplikacije on je najpopularniji. Korištenje *Python*-a u razvojnom okruženju *Django,* ima svoje prednosti, jedna od njih je što *Django* čini web stranicu bržom i jednostavnijom. Da bi se koristio, *Django* se mora instalirati dodatno uz *Python* kako bi se mogle koristiti biblioteke odnosno paketi koje *Django* sadrži. Slika 6 prikazuje jednostavan kôd koji se šalje u *HTML* datoteku, te mijenja sadržaj pod određenim elementom.

```
from django.shortcuts import render_to_respone

from blog.models import posts

def home(request):
    content = {
        'naslov' : 'Moj prvi post',
        'autor' : 'Prezime',
        'datum' : '18.srpanj.2017',
        'body' : 'Lorem ipsum dolor sit amet.'
    }
    return render_to_respone('index.html', content)
```

Slika 6. Primjer Python-a

Slika 7 prikazuje *HTML* kôd koji se prikazuje kao stranica, *HTML* dokument raspoznaje *Python* po oznakama „{{ }}“, sve što se nalazi unutar oznaka može koristiti kao *Python-ov* kôd.

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>Test</title>
    </head>
    <body>
        <h1>Prvi blog</h1>
        <h2>{{ naslov }}</h2>
        <h3>Postano u {{ datum }} od {{ author }}</h3>
        <p>{{ body }}</p>
    </body>

</html>
```

Slika 7. Prikaz HTML kôda uz Python

Iz Slike 7 se vidi kako unutar *HTML* oznake se nalazi varijabla od *Python*-a. Kada se pogleda Slika 6 i Slika 7, može se zaključiti da će unutar oznaka „<h2></h2>“ biti prikazan sadržaj „Moj prvi post“, te isti slučaj će se primijeniti na ostalim varijablama.

7. PERL

Ovo poglavlje je posvećeno skriptnom jeziku nazvanom *Perl*. Govorit će se o njegovoj funkciji, težini kao skriptnog jezika, razlog njegove popularnosti i pogodnosti korištenja.

Perl je implementiran kao jezik koji koristi interpreter. Kratica *Perl* znači *Practical Extraction and Report Language*, također kratica se koristi i za *Pathologically Eclectic Rubbish Lister*. *Perl* je srodan s visokom razinom programskog jezika, te je program opće namjene i interpreter dinamičkih programskih jezika. Kreator jezika *Perl* je Larry Wall 1987. godine, kreiran je kao jezik opće namjene za *Unix* operacijski sustav koji je predstavljao skriptni jezik da olakša obradu izvješća. *Perl* jezik je uzimao neke ideje i značajke od drugih programskih jezika kao što je *C, shell script, AWK* i *sed*. Moguće je susresti se s nazivom *perl* koji u sebi sadrži malo početno slovo, on predstavlja interpreter kojeg pokreće program dok se *Perl* s velikom odnosi na sam jezik. Larry je pokušavao izraditi izvješća iz *Usenet*-ovih vijesti koja je prikupljala datoteke vezana za izvješća, koja su se odnosila na greške u sustavu. Larry je odlučio prekoračiti problem pomoću alata za opću namjenu koji bi se mogao koristiti i na drugim mjestima. Rezultat je bio *Perl* verzija nula (Schwartz, 2007).

Jedna od prednosti *Perl* skriptnog jezika je što sintaksa sliči *shell* jeziku. Također, prednost je u zrelosti *Perl* jezika, što znači da je svestran. Ovisno o zahtjevima i načinu programiranja, *Perl* može biti proceduralan, objektno orijentiran ili funkcionalan. Može biti izvrsna zamjena za *sed* ili *awk*, jer ima sličnu sintaksu, ali s više funkcionalnosti. Još jedna prednost *Perl*-a je što upravlja datotekama. Nedostaci *Perl*-a su sljedeći, njegova sintaksa se ne sviđa mnogim programerima zbog toga što u usporedbi s drugim skriptnim jezicima, *Perl* ima loš faktor upotrebe. Ne predlaže se kao odabir jezika za početnike. *Perl* je u mnogim zadacima skriptiranja spor. Isti rezultat može se postići uz nekoliko načina što čini kôd neurednim (Schwartz, 2007).

7.1. Da li je Perl težak jezik ?

Prema Schwartz-ovoj knjizi (2007) *Perl* nije težak za korištenje, no ponekad je težak za naučiti, generalizirao je. Jezik je stvoren tako da se pokušava prilagoditi obliku na koji ljudi razmišljaju o određenim problemima, te ne pruža ništa protivno ljudskim očekivanjima. Dizajneri *Perl*-a vjeruju da je *Perl* popularan jezik, ne samo za matematičare i računalne znanstvenike. Postoje mnogi ljudi koji nisi znanstvenici, a u većini slučajeva uspješno koriste *Perl* (Schwartz, 2007).

Perl ne želi da programer vidi stvari na način na koji računalo vidi. Umjesto toga, *Perl* programeru omogućuje razvitak svog osobnog pristupa programiranju. Postoji više od jednog načina da se nešto napravi, a *Perl* omogućuje programeru da programira na način koji on želi ili način na koji je zamišljen. Ako je programer barem jednom napisao *C* program, *awk skriptu*, *shell* skriptu ili *BASIC* program, onda njegovo znanje pospješuje znanju *Perl* jezika, te programer onda zna većinu stvari o njemu. Većina zadataka zahtijeva samo mali dio jezika *Perl*. Postoji više načina za izvršavanje određenog zadatka (Schwartz, 2007).

Perl je jezik koji koristi interpreter. To znači da se može pisati vlastiti program u kratkom vremenu, te ga se može testirati bez prethodnog koraka za kompiliranje što omogućuje brzo i jednostavno testiranje te uklanjanje grešaka. Da bi se *Perl* mogao lakše naučiti treba imati iskustva u *Unix*-u, gotovo bilo kakvo programsko iskustvo, razumijevanje regularnih izraza i sposobnost razumijevanja kôdova drugih ljudi pospješuje znanju *Perl*-a.

7.2. Popularnost Perl-a

Nakon što se je Larry usavršio *Perl*, objavio ga je zajednici čitatelja *Usenet*, poznat pod imenom „*the Net*“. Korisnici širom svijeta dali su povratne informacije, ispitivali ga na koji je način to napravio i mnoge druge stvari. No, kao rezultat, *Perl*-ova populacija je počela svakim danom samo rasti. Larry danas ne piše kôd, ali on i dalje vodi razvoj i donosi velike odluke. *Perl* uglavnom održava grupu ljudi koji se zovu *Perl 5 Porters*. Nekoć je *Perl* bio mali jezik dostupan samo na nekoliko *Unix* sustava, a sada sadrži tisuće besplatnih stranica o dokumentaciji *Perl*-a, desetke knjiga, nekoliko glavnih *Usenet* grupa s nebrojenim čitateljima i implementacija na gotovo svakom sustavu koji se danas koristi (Schwartz, 2007).

Danas se posjeduje verzija *Perl 6*, no *Perl 5* verzija se i dalje čuva kod većine, jer je i dalje aktualna i stabilna inačica. U 2017. godini *Perl* je i dalje popularan na mnogim stranicama te ga svrstavaju među petnaest najpopularnijih jezika današnjice.

7.3. Primjena Perl-a

Perl je poželjno koristiti za male programe koji se mogu razviti unutar nekoliko minuta. Također, može se koristiti za duge i opsežne programe za koje je potrebno nekoliko desetaka programera kojima je potrebno nekoliko godina da završe projekt. Isto tako, postoje programi koji se mogu sadržavati od nekoliko sati programiranja od samog početka tj. stvaranja plana, do potpunog testiranja kôda.

Perl je jezik koji se može koristiti za veliki broj zadataka. Ukratko, tipična uporaba bila bi vađenje podataka iz tekstualne datoteke, te ispisivanja izvješća ili za pretvaranje tekstualne datoteke u drugi oblik. No, *Perl* pruža veliki broj alata za prilično složene probleme, uključujući programiranje sustava. Programi pisani u *Perl*-u nazivaju se *Perl* skripte dok se izraz *perl* odnosi na program sustava za izvršavanje *Perl* skripti. Ako programer koristi *shell* skripte, *awk* ili bilo koje *Unix* uslužne programe za razne svrhe, naići će na mogućnost korištenja *Perl*-a za te programe te za mnoge druge svrhe. No ako programer nije koristio takve alate, ali počeo je razmišljati da ima potrebu vezati

se za *Perl* program, onda ono što stvarno treba učiniti je opredijeliti se za *Perl* umjesto ostalim bespomoćnim jezicima (Schwartz, 2007).

Perl je optimiziran za probleme koji oko 90% rade s tekstom, a ostalih 10% je sve ostalo. Taj opis čini se kao da odgovara većini programskih zadataka koji se pojavljuju ovih dana. U savršenom svijetu, svaki programer znao bi svaki jezik, uvijek bi postojao izbor nekoliko jezika za svoj projekt, u većini slučajeva to bi bio *Perl*.

Također, *Perl* je jezik za *CGI* skripte, mnogi programeri koji tek ulaze u svijet programiranja postavljaju pitanje „Nije li *CGI* samo *Perl*?“ ili „Za što se sve može koristiti *Perl* osim za *CGI* skripte?“. *Perl*-ov utjecaj ne osjeća se među *shell* skriptama. Ne samo da može koristiti datoteke, on također ima veliku važnost u svijetu *CGI* skripti na *World Wide Web*-u. Pronalazi se mnogo automatiziranja komunikacije između poslužitelja i preglednika širom svijeta u različitim oblicima koji su napisani u *Perl*-u. *Perlscript* je *Perl* skripta koja se može izvoditi na klijentskoj i poslužiteljskoj strani. *Perl*-ova glavna funkcija na webu je način na koji će se *CGI* skripte izvoditi (Schwartz, 2007).

Perl se koristi u mnogo područja, jedno od tih područja su web stranice ili web aplikacije. *Perl* ima nekoliko razvojnih okruženja za web stranice kao što je *Catalyst*, *Jifty*, *Mojo* i *Titanium*. Koristi se i u konfiguracijskom menadžmentu, administraciji sustava, bio-informatici, razvoju igara, te u mnogim drugim područjima i aplikacijama.

Na Slici 8 se može vidjeti jednostavan program napisan u skriptnom jeziku *Perl*, unutar skriptnog jezika se nalaze linije kôda koje se prikazuju na web stranici. *Perl* je jezik koji se izvršava na strani poslužitelja tako da se skripta šalje na server, te se na serveru obrađuje i vraća klijentu. Kada korisnik upiše *URL* gdje se nalazi skripta od *Perl*-a, dobiti će sadržaj na kojemu će pisati „Hello, world!“.

```
#!/usr/local/bin/perl.exe

print "Content-Type: text/html\n\n";
# Komentari

# HTML kod koji slijedi

print "<html> <head>\n";
print "<title>Hello, world!</title>";
print "</head>\n";
print "<body>\n";
print "<h1>Hello, world!</h1>\n";
print "</body> </html>\n";
```

Slika 8. Primjer Perl programa

8. USPOREDBA SKRIPTNIH JEZIKA

U ovom poglavlju usporedit će se skriptni jezici, usporedba će biti po kriteriju jednostavnosti sintakse te jednostavnosti samog jezika, isto tako će se spomenuti prednosti i nedostaci te odnosi između pojedinih jezika. Na kraju poglavlja će biti prikazan popis rangiranih skriptnih jezika po kriteriju jednostavnosti sintakse i jednostavnosti jezika. U usporedbi su korišteni skriptni jezici koji se nalaze u završnom radu. Koristit će se isti primjeri za svaki skriptni jezik kako bi se mogla usporediti sintaksa skriptnih jezika, primjeri su korišteni iz rada koji je napravljen od strane Lovrenčić et al. (2009) gdje svaki primjer traži najveću vrijednost odnosno broj u danom nizu te ga na kraju ispisuje.

JavaScript je jezik koji koristi kombinaciju funkcija *C*, *C++* i *Jave*, tako da je programeru omogućen odabir najboljeg jezika. *JavaScript* je rangiran među najkreativnijim jezicima današnjice.

Razlika između *JavaScript*-a i *PHP*-a je velika. *JavaScript* je skriptni jezik koji se izvršava na strani klijenta, dok se *PHP* izvršava na strani poslužitelja. Zajedno tvore jednu dinamičnu web stranicu. Obzirom da je *JavaScript* jezik koji se izvršava na strani klijenta, a *PHP* na strani poslužitelja, postoje neke sličnosti koje valja spomenuti. Jedna od sličnosti između *JavaScript*-a i *PHP*-a je što oba jezika koriste interpreter, te spadaju u kategoriju skripata koji se preporučuju početnicima, prisutni su svugdje te ih sadrži gotovo svaka web stranica.

JavaScript je sličan *Javi*, ali ga je lakše koristiti. Može se koristiti kao proceduralni i objektno orijentirani jezik. *JavaScript* je danas standard kada govorimo o dinamičnosti na web stranicama i skriptiranju na strani klijenta. Ima sličnu sintaksu kao *C* programski jezik, no njegova namjena je sasvim drugačija. *JavaScript* je brz i moćan, budući da je na klijentskoj strani, no uvijek postoji sigurnosni problem. Danas postoji mnogo mogućih rješenja koja se koriste za skrivanje *JavaScript* kôda, jedno od rješenja je kombiniranje *JavaScript* kôda s kôdom od *PHP*-a, tada korisnik neće biti u mogućnosti vidjeti *JavaScript* kôd (Lovrenčić et al., 2009).

Smatra se da je *JavaScript* najbolji skriptni jezik za programiranje na strani klijenta te da je *PHP* najpopularniji za skriptiranje na strani poslužitelja. *JavaScript* može

onemogućiti preglednik, dok *PHP* ne može. *PHP* se primjenjuje pri dohvaćanju podatka iz baze te kada je u pitanju sigurnost na stranici, dok namjena *JavaScript*-a je poboljšanje sučelja web stranice. Korištenje *JavaScript*-a je jednostavnije od korištenja *PHP*-a. Sintaksa kod *PHP*-a i *JavaScript*-a ja vrlo jednostavna u usporedbi s drugim jezicima, no mnogi tvrde kako je *JavaScript* sintaksa jednostavnija od *PHP*-ove. Sintaksa od *JavaScript*-a se može vidjeti na Slici 9.

```
var x=new Array(3,4,5,6);
var j=0;
for(i=0;i<x.length;i++)
{
      if(x[i]>x[j])
      j=i;
}
alert(x[j]);
```

Slika 9. JavaScript sintaksa (Lovrenčić et al. 2009, p. 141)

PHP-ova sintaksa je slična *Perl*-ovoj sintaksi, razlog tome je što je *PHP* preuzeo veliku količinu sintakse od *Perl*-a. Kada je *PHP* prvi put kreiran, bio je alternativa *Perl*-u, što je omogućilo lakši način dinamičkog skriptiranja. *PHP* je vrlo jednostavan za primjenu. Danas ima mnogo *PHP* interpretera koji se izvršavaju na različitim platformama kao što je *Windows, Unix, Linux* i *Mac*. *PHP* se razvija velikom brzinom u usporedbi s drugim razvojnim jezicima, kao što je *Java* odnosno njegova platforma za razvoj web aplikacija *JSP*. *PHP* je objektno orijentiran i proceduralan, što programerima omogućuje odabir bilo kojeg načina implementacije. Omogućuje stvaranje skriptnih klasa ili funkcija korištenjem sintakse kao što koristi *C++* ili *Java*. Kako *PHP* ima nizak stupanj tipiziranja, programer ne treba brinuti što će funkcija unutar neke kase vratiti, dok u *C++* programer treba voditi računa o povratnom tipu jer će kompajler javiti grešku. *PHP* je besplatan, jednostavan, brz i opremljen mnogim funkcijama za razne svrhe što ga čini vrlo pogodnim za web aplikacije (Lovrenčić et al., 2009).

Kada se usporedi *PHP* s *Python*-om, *PHP* je skriptni jezik koji se ugrađuje direktno u sadržaj dokumenta. Jednostavan primjer korištenja *PHP*-a unutar *HTML*-a se može vidjeti na Slici 4 gdje se ispisuje tekst korisniku na ekran. Vrlo je koristan pri manjim web stranicama, no kako se internet razvija aplikacije postaju sve kompleksnije. *PHP* kôd koji se ugrađuje unutar *HTML* oznaka se prebacuje u predložak odnosno u posebne datoteke koje služe za prikaz, dok je kôd poslovne logike posebno prebačen u druge datoteke radi bolje jasnoće i održavanja. Još jedan *PHP* primjer koji je prikazan Slikom 10, gdje su prikazani osnovni jezični konstrukti te se sintaksa može usporediti s ostalim skriptnim jezicima koji su u ovom poglavlju.

```
<?php
function max($a)
{
  $maxVal = $a[0];
  for($i = 1; $i < count($a); $i++)
  {
    if ($a[$i] > $maxVal)
    {
      $maxVal = $a[$i];
    }
  }
  return $maxVal;
}
?>
```

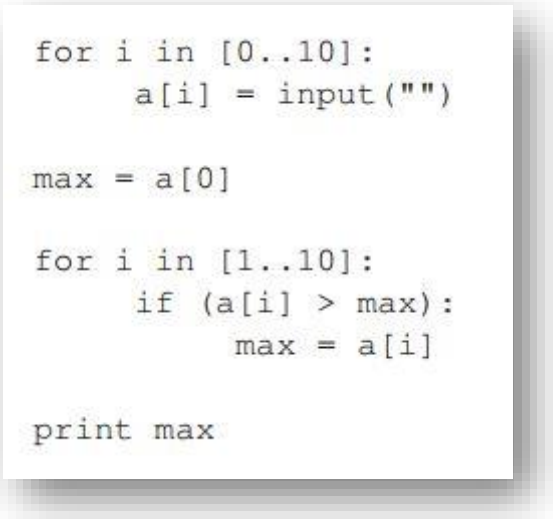
Slika 10. PHP sintaksa (Lovrenčić et al. 2009, p. 139)

Jedan od nedostataka *PHP*-a je integracija baze podataka direktno u kôd, što je dugoročno gledano nedostatak *PHP*-a, jer je tada baza podataka čvrsto povezana s *PHP-ovim* određenim funkcijama. *PHP*-ova sintaksa je slična sintaksi *C*, varijabla se mora deklarirati sa znakom „$“ i treba biti odmah inicijalizirana, pristup globalnoj varijabli može biti omogućen pomoću ključne riječi „global“ (Purer, 2009).

Python je poznat po kôdu koji je čitljiv te kôd koji ima jednostavnu sintaksu, koja se može lagano naučiti. *Python* je bio inspiriran *ABC* programskim jezikom te s nekoliko značajki koje dolaze od programskog jezika *C*. *Python*-ov kôd je čitljiv i

jednostavan, sintaksa je prilično slična *BASIC*-ovoj. Njegova je sintaksa i semantika svedena na minimum, dok je standardna biblioteka jako velika. Za razliku od drugih jezika, niti jedan konstrukt u *Python*-u ne mora imati eksplicitnu oznaku za kraj odnosno oznaka „;“. *Python* se može proširiti s *C* ili *C*++, takva radnja se primjenjuje kada je potrebna vremenski kritička primjena. Najveća prednost je izražajna i čitljiva sintaksa, što znači da se s vrlo kratkim kôdom može učiniti mnogo. *Python* je relativno spor pri izvršavanju u usporedbi s programskim jezikom kao što je *C* (Lovrenčić et al., 2009).

Python je skriptni jezik koji nije fokusiran samo na web aplikacije, stoga je trebalo neko vrijeme da *Python* postane značajan za web. Prvi pristup je bio pomoću *CGI* skripti. Za razliku od *PHP*-a, *Python* je od početka bio objektno orijentirani jezik, no nije ograničen na paradigmu objektno orijentiranog jezika, također podržava proceduralno programiranje i neke funkcionalne značajke. U usporedbi s ostalim skriptnim jezicima u završno radu *Python* je najjednostavniji skriptni jezik za korištenje, te ima najjednostavniju sintaksu. *Python*-ova ideja tabulatora umjesto zagrada u kôdu možda izgleda sjajno, ali to je ozbiljan nedostatak pri uklanjanju pogrešaka programa (Purer, 2009).



```
for i in [0..10]:
     a[i] = input("")

max = a[0]

for i in [1..10]:
     if (a[i] > max):
          max = a[i]

print max
```

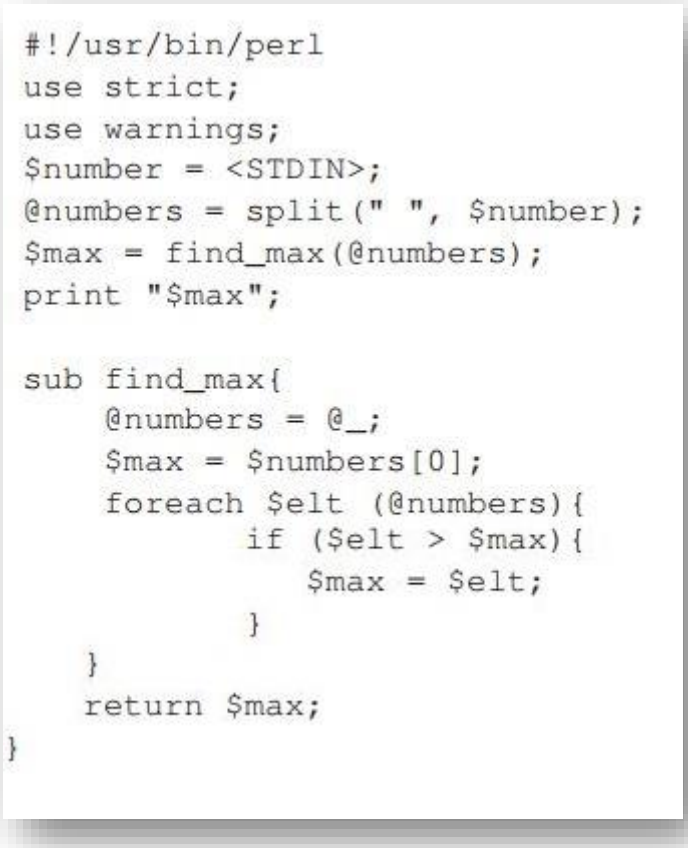
Slika 11. Sintaksa Python-a (Lovrenčić et al. 2009, p. 126)

Slika 11 prikazuje program koja služi za pronalazak najvećeg broja u upisanom nizu. U usporedbi s *PHP* kôdom, vide se razlike, *Python*-ova sintaksa je lagana za čitati i razumjeti, jer postoji samo mali skup ključnih riječi. Razlikuje se od *PHP*-a i *Perl*-

a po jednom velikom svojstvu, a to je obavezno uvlačenje blokova unutar kôda. To znači da je izvorni kôd uvijek pravilno strukturiran, skriptni jezik poput *JavaScript, PHP* i *Perl*-a to ne provodi. Kôd napisan u *Python*-u ne treba biti odvojen s točka zarezom, jer nova linija kôda definira kraj prethodne, no nije zabranjeno. Može se vidjeti iz primjera da se varijabla inicijalizira samo s nazivom te se dodijeli vrijednost, nije potrebno kao u *PHP*-u ili *Perl*-u dodavati ispred varijable znak „$“ ili kao u *JavaScript*-u dodavati oznaku „var“.

Perl je potpuni jezik sa svim jezičnim elementima, ima kontrolu strukture i potprograma. Izvorno je napisan da manipulira tekstom u datotekama, izvuče podatke iz datoteka i piše izvještaje, no kroz kontinuirani razvoj može manipulirati i procesima, obavljati poslove umrežavanja, obrađivati web stranice, komunicirati s bazom podataka, analizirati znanstvene podatke, te mnogo drugih stvari, pod uvjetom da je vezan za različite biblioteke. Varijable u *Perl*-u nisu, kao i u drugim programskim jezicima deklarirane određenom tipu podataka. Vrsta podataka određene varijable definirana je posebnim znakom na početku varijable „$“, dok se za polja koristi znak „@“. Pokazivači u *Perl*-u rade na *Pascal*-ov način, koji je mnogo sigurniji od *C* ili *C++.* Rad s datotekama u Perl-u izgleda kao i kod *C++*. No, pokušavajući dodati mnogo dobrih svojstva, Perl postaje velik, spor i nečitljiv jezik koji ne brine o optimalnosti i brzini kôda (Purer, 2009).

Perl i *Python* se koriste za slične svrhe, no način djelovanja im je različit. Kada se uzme kriterij jasnoće i fleksibilnosti sintakse, *Python* koristi uvlačenje funkcija i kôda kako bi program bio čitljiviji, to je dobra praksa i kod drugih jezika, no *Python* jednostavno neće raditi ako se ne koristi ispravno uvlačenje kôda. *Perl* ima tradicionalni pristup, pomoću vitičastih zagrada za tijelo funkcije odnosno petlje, tada je programer slobodan pisati svoj kôd po vlastitoj želji, bez pravila uvlačenja. Kada se uzme u obzir upotrebljivost i snaga, *Perl* se mnogo dulje koristi od *Python*-a, stoga puno više materijala za ponovnu upotrebu ima *Perl* što ga čini iznimno moćnim. No, *Python* je bolji odabir za novog programera koji želji izvršiti jednostavan zadatak. *Python*-ove naredbe i ključne riječi intuitivno su povezane s njihovim funkcijama, što čini jezik jednostavnijim za koristiti. Nakon što programer nauči osnovu *Python*-a, može brzo napredovati u pisanju programa koji je koristan i zabavan.



```
#!/usr/bin/perl
use strict;
use warnings;
$number = <STDIN>;
@numbers = split(" ", $number);
$max = find_max(@numbers);
print "$max";

sub find_max{
    @numbers = @_;
    $max = $numbers[0];
    foreach $elt (@numbers){
            if ($elt > $max){
                $max = $elt;
            }
    }
    return $max;
}
```

Slika 12. Funkcija za ispis Perl (Lovrenčić et al. 2009, pp. 121-122)

Da bi se *Perl* izvršio, svaki *Perl* program mora proći kroz *Perl*-ov interpreter. *Perl*-ov interpreter se obično instalira u usr/bin datoteku na lokalnom disku, iz tog razloga *Perl* program počinje linijom „#! / Usr / bin / perl“. Slika 12 prikazuje funkciju koja pronalazi najveći element u danom nizu. Na Slici 12 se vidi sintaksa *Perl* skriptnog jezika. *Python* za razliku od *Perl*-a, ne koristi vitičaste zagrade te je potrebno uvlačiti blokove kôdova, dok kod *Perl*-a to nije slučaj. *Perl* ima sličnu sintaksu kao *C*, kao što je vidljivo iz slike.

Na kraju može se donijeti zaključak da je *Python* najjednostavniji skriptni jezik za korištenje, te ima najjednostavniju sintaksu. Po kriteriju jednostavnosti sintakse i jezika, skriptni jezici rangiraju se sljedećim redoslijedom:

1. *Python*
2. *JavaScript*
3. *PHP*

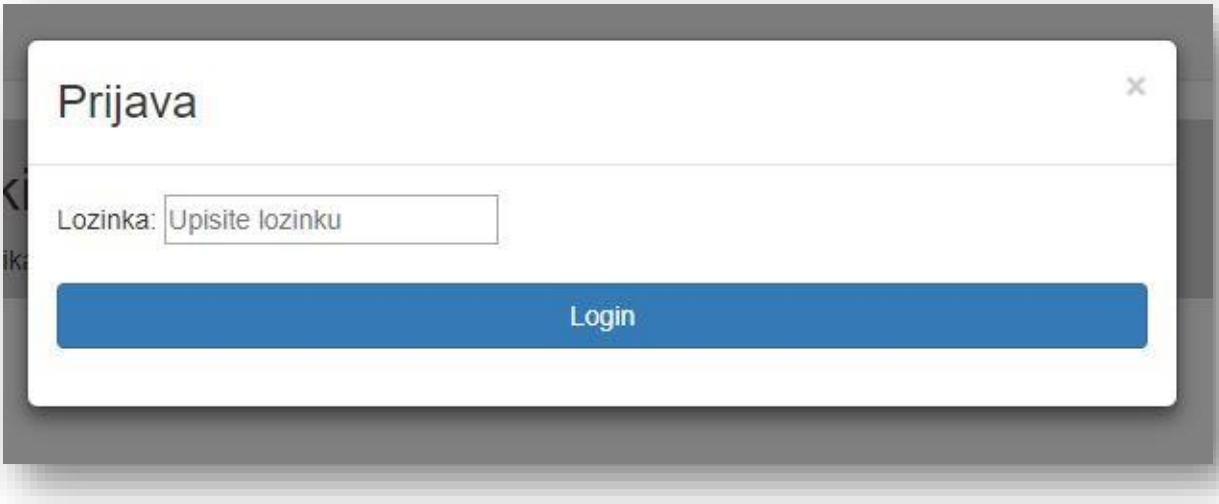
4. *Perl* (Denisco, 2017).

9. PRAKTIČNI DIO

Kako bi se bolje razumio koncept skriptnih jezika i na koji način rade, pripremljen je praktični dio, gdje su korištena dva skriptna jezika kao primjer, kako bi se lakše shvatila funkcija lijepljenja skripti. Za ovaj primjer koriste se dva skriptna jezika, jedan se izvršava na klijentskoj strani, a drugi na strani servera odnosno na poslužiteljskoj strani. Na klijentskoj strani je *JavaScript*, a na strani poslužitelja se nalazi *PHP*. Kako bi se mogao koristiti primjer i kako bi mogao biti testiran, datoteke praktičnog dijela moraju biti na poslužitelju, tako da se *PHP* može izvršiti, u protivnom će se dobiti neupotrebljiva stranica koja će prikazati nejasne kôdove koji upućuju da se *PHP* ne može izvršiti jer nema poslužitelja na kojemu se on izvršava. Također je potrebno kreirati bazu podataka te se u nju mora ubaciti datoteka „ispiti.sql“ kako bi program mogao ispravno raditi i dobivati potrebne podatke koje se prikazuju na stranici.

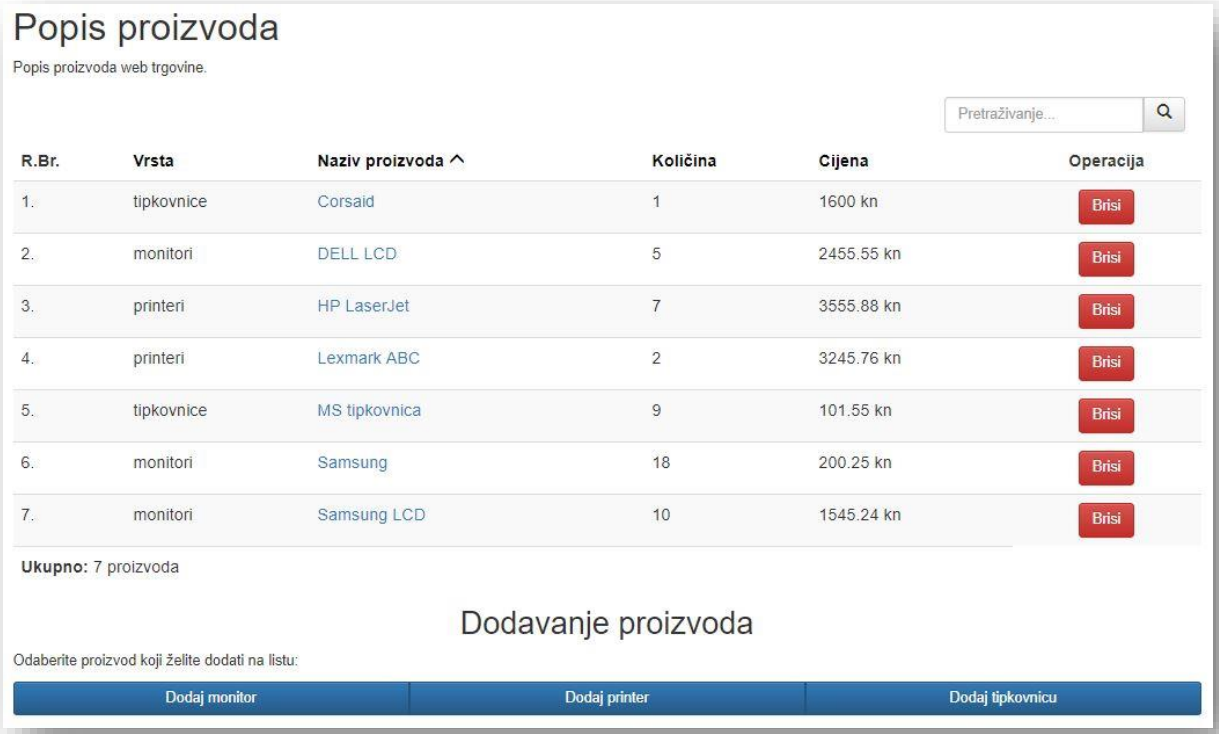
U ovom primjeru nalazi se web stranica za evidentiranje proizvoda u skladištu. U skladištu se evidentiraju tri stvari: tipkovnice, monitori i printeri. Primjer sadrži četiri *PHP* datoteke, svaka od njih će se ukratko objasniti, radi boljeg snalaženja unutar primjera korišteni su komentari u kôdu.

Prva stranica s kojom se korisnik susreće je *login* forma odnosno *PHP* datoteka „login.php“. Forma sadrži polje za lozinku, lozinka mora biti sadržaja „abc“. Forma se uzima preko *PHP*-a te se provjerava na serveru, ali u ovom slučaju ne na bazi nego samo *if* uvjet koji vraća rezultat ispravnosti lozinke. Prikaz stranice nalazi se na Slici 13.



Slika 13. Stranica koju prikazuje login.php

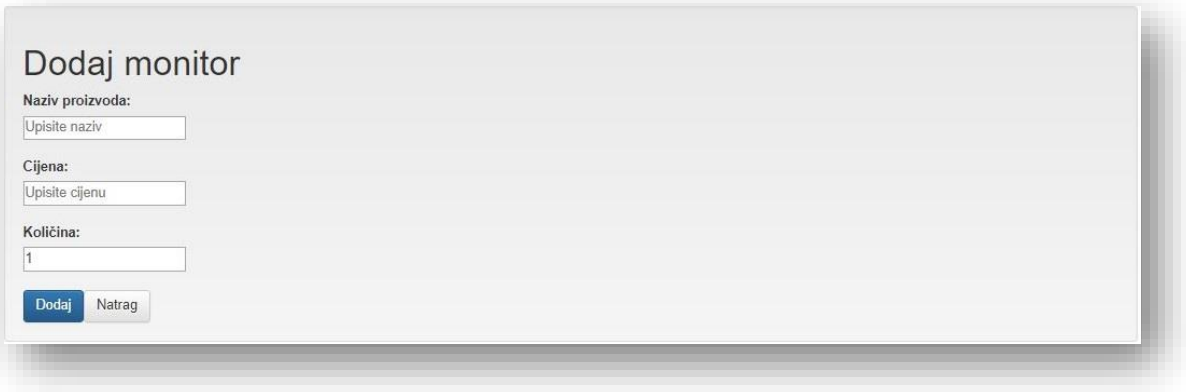
Kada je korisnik došao preko forme, dolazi do stranice koja prikazuje popis proizvoda odnosno *PHP* datoteku „popis.php“. U slučaju da je korisnik izravno upisao link za datoteku „popis.php“, web stranica ga vraća na *login* formu.



Slika 14. Stranica koju prikazuje popis.php

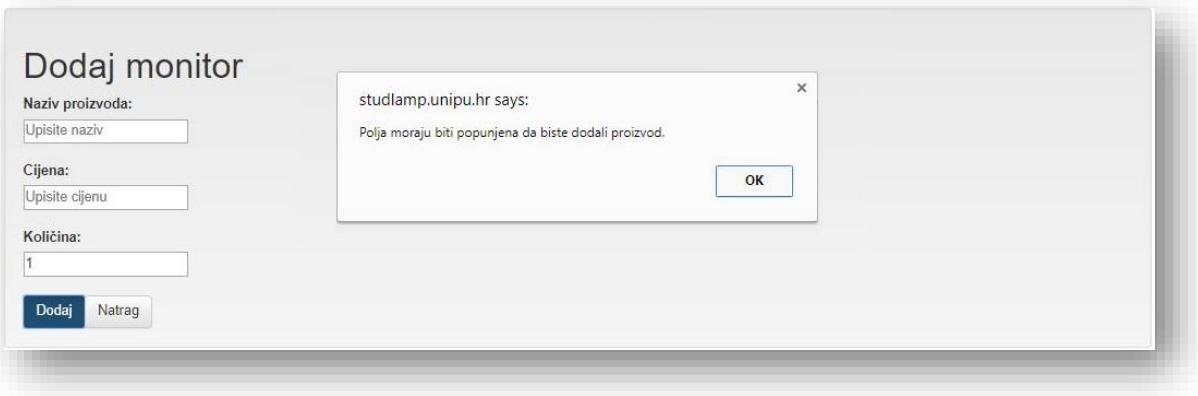
Nakon što je korisnik ispravno ispunio *login* formu, prikazuje mu se tablica koja je učitana iz baze podataka. Podaci koji se nalaze u tablici dohvaćeni su pomoću *PHP*-a. *PHP* je poslao upit serveru za dohvat podataka iz baze te ih prikazao u *HTML* obliku koji se prikazuje korisniku. Na Slici 14. se može vidjeti nekoliko funkcionalnosti, kao što je pretraživanje proizvoda po nazivu te sortiranje proizvoda po vrsti, nazivu, cijeni i količini. Praktični dio sadrži u sebi *JQuery* koji predstavlja biblioteku *JavaScript*-a. Pomoću *JQuery*-a web stranica je interaktivnija i korisniku zanimljivija. Dolaskom na stranicu „popis.php“ korisnik može vidjeti animirani prikaz tablice te prolaskom kursora kroz proizvode odnosno redove, *JavaScript* mijenja dizajn aktivnog reda. Vidljivo je da na nazivima proizvoda postoje linkovi koji služe za izmjenu pojedinog artikla, dok se u zadnjem stupcu nalazi dugme „Briši“ pomoću kojeg korisnik briše određeni proizvod iz baze podataka. Ako korisnik klikne na link naziva proizvoda ili na dugme dodaj za bilo koji proizvod, preglednik će učitati stranicu za dodavanje ili izmjenjivanje, ovisno o tome što korisnik odabere.

Kada se učita *PHP* datoteka „dodaj.php“, korisnik dobije formu za unos naziva i cijene. *JavaScript* pomoću biblioteke *JQuery* prikazuje formu na interaktivan način korisniku. Izgled stranice se može vidjeti na Slici 15. Ako korisnik unese proizvod pod istim nazivom, kategorijom i cijenom, količina će se samo nadodati na postojeći proizvod.



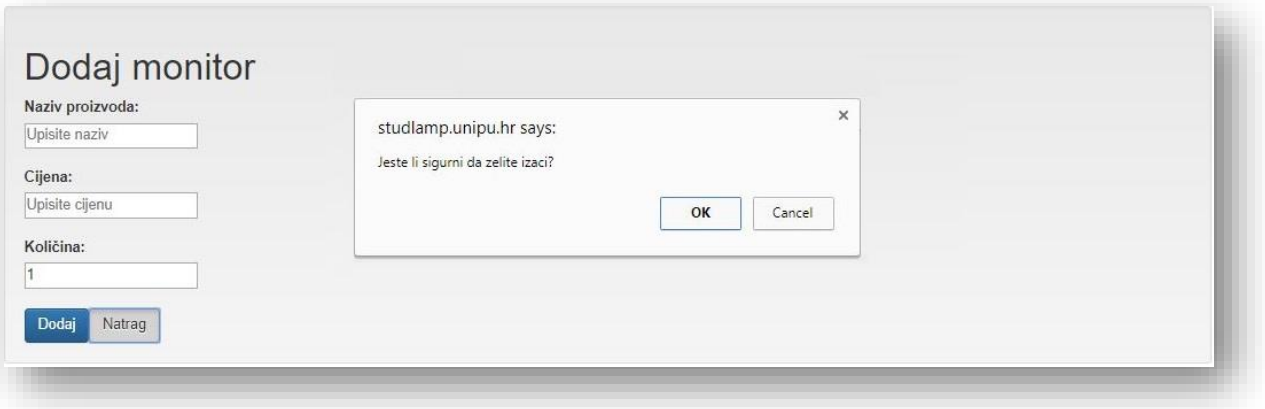
Slika 15. Stranica koju prikazuje dodaj.php

Klikom na dugme dodaj monitor korisnik dobiva stranicu za dodavanje monitora. Ako klikne na dugme dodaj, stranica će korisnika upozoriti da sva polja moraju biti unesena, primjer se može vidjeti na Slici 16. Upozorenje koje se prikazuje korisniku je napisano u *JavaScript*-u.



Slika 16. JavaScript upozorenje za dugme Dodaj

Ako korisnik klikne na dugme natrag, stranica će mu poslati upit da li je siguran da želi izaći iz forme. Primjer se može vidjeti na Slici 17. Ukoliko klikne na dugme „OK“, izlazi iz trenutne stranice te ga stranica vraća na popis proizvoda odnosno na *PHP* datoteku „popisi.php“. Ukoliko odabere dugme „cancel“ obustavlja se akcija te korisnik ostaje na stranici.



Slika 17. JavaScript upozorenje za dugme Natrag

Ovaj primjer prikazuje kako dva skriptna jezika mogu funkcionirati zajedno. Dok se jedan skriptni jezik izvršava na klijentskoj strani, drugi se izvršava na strani poslužitelja.

Kombiniranjem više skriptnih jezika može se napraviti aplikacija po želji s različitim mogućnostima. Skriptni jezici su demonstrirani primjerom, te se može zaključiti iz primjera koja je funkcija skriptnog jezika, te zašto se skriptni jezik naziva jezikom lijepljenja (jer se mogu kombinirati dva skriptna jezika, odnosno lijepiti se jedan za drugoga).

Izrađivanje *Python* i *Perl* skripte nije potrebno, jer se pomoću dva popularna skriptna jezika u ovom primjeru prikazana je moć skriptiranja, te kako funkcionira lijepljenje skripata.

Četvrta datoteka je „inc_conn.php“ koja sadrži podatke o konekciji na bazu podataka, sve što korisnik treba je nabaviti domaćina odnosno poslužitelja na kojeg će se staviti baza podataka te učitati datoteke koje se nalaze u ovom primjeru.

Primjer zadatka i datoteke može se preuzeti na linku:
https://www.dropbox.com/s/xu8hx8qxpkqdcgz/Skriptni%20programski%20jezici.zip?dl=0

10. ZAKLJUČAK

Skriptni jezici čine web mjesto lakšim za korištenje, bržim, te na najbolji način iskorištavaju resurse. Zaključuje se kako je skriptni jezik lakše savladati nego programski jezik. Izrađivanje aplikacije danas pomoću skriptnog jezika je brže nego pomoću programskog jezika. Aplikacija se može izraditi brže zbog toga što je lakše uočiti grešku kod skriptnog jezika, te se može ispraviti uvijek na vrijeme što samom programeru uštedi puno vremena.

Razvoj skriptnog jezika je danas dosegao do položaja gdje on nije ništa manje bitniji od sistemskog programskog jezika. Svakim danom se razlika između dvaju tipova navedenih jezika smanjuje. Također sistemski jezik napreduje zato što uključuje elemente skriptnog jezika.

Skriptni jezici nisu bili toliko popularni sve do početka devedesetih godina kad su se razvili skriptni jezici poput *JavaScript, PHP-a i Python*-a. Do tada su sistemski jezici bili u centru pažnje, a *Perl* je tada održavao Web zajednicu.

Uporaba skriptnih jezika svakim danom sve više raste. Danas je računalo snažnije nego što je bilo prije desetak godina, tako da skriptni jezici također postaju popularni uz jačanje računalnih komponenti. Potražnja za skriptnim jezicima je sve veća, no to ne mora značiti da će programski jezici nestati i da će se samo upotrebljavati skriptni jezici.

Skriptni jezik predstavlja drugačiji skup programiranja od jezika za programiranje sustava. Ne stavlja fokus na brzinu izvršavanja i snagu programskih jezika, nego pruža znatno veću produktivnost programu, te ponovnu upotrebu softvera. Programski jezik sustava je prikladan za izgradnju komponenti gdje je složenost u strukturama podataka i algoritmima dok je skriptni jezik prikladan za lijepljenje aplikacija gdje se njegova složenost nalazi u vezi.

Program pisani u skriptnom jeziku razumno je jednostavan za pisanje. Jedna od glavnih prednosti kod skriptnih jezika je održavanje, kôd je dizajniran da se može lagano čitati, programer može lagano nakon određenog vremena proučiti kôd i shvatiti za što je program namijenjen.

U završnom radu navedeni su najpopularniji današnji skriptni jezici. Svaki od njih je pojedinačno obrađen. Kako bi se napravila kompletna i profesionalna aplikacija mora se koristiti i kombinirati više skriptnih jezika što je moguće vidjeti u primjeru koji se nalazu u završnom radu. Gotovo svaka današnja web stranica sadrži *JavaScript* u sebi, kao i *PHP* skriptni jezik. Danas jedan od najpoznatijih skriptnih jezika među ostalima je *JavaScript*, može se koristiti za izrađivanje aplikacija skoro bilo kakvog tipa. Mnogo drugih skriptnih jezika je dostupno danas. Zadatak lijepljenja skripata postaje sve češći tako da će skriptiranje postati još važnija programska paradigma u budućnosti. Svaki od navedenih jezika može mnogo toga ponuditi te je ključno da se odaberu oni pravi.

11. LITERATURA

1. Bulger et al. (2014), *MySQL/PHP Database Applications, 2nd ed.*,<raspoloživo na: http://download.pattayacitythailand.net/Download/PDF/PHP%20books/Wiley%20MySQL%20PHP%20Database%20Applications%202nd.pdf >, [pristupljeno 10.07.2017].

2. Codeconquest (2017), *Client Side vs Server Side,* [Internet], <raspoloživo na: http://www.codeconquest.com/website/client-side-vs-server-side/ >, [pristupljeno 12.07.2017].

3. Crockford D. (2008), *JavaScript: The Good Part, O'Reilly Media*, Sebastopol, [pristupljeno 11.07.2017].

4. Denisco A. (2017), *The 10 easiest programming languages to learn,* [Internet], <raspoloživo na: http://www.techrepublic.com/article/the-10-easiest-programming-languages-to-learn/ >, [pristupljeno: 24.08.2017].

5. Gilmore W. J. (2001), *PHP History,* [Internet], <raspoloživo na: http://about.brighton.ac.uk/is/aranea/php4book.pdf >, [pristupljeno 13.07.2017].

6. Holigan A. (2016), *Five new features PHP 7,* [Internet], <raspoloživo na: http://blog.teamtreehouse.com/5-new-features-php-7 >, [pristupljeno 15.08.2017].

7. Kanavin A. (2002), *An overview of scripting languages*, <raspoloživo na: http://www.sensi.org/~ak/impit/studies/report.pdf >, [pristupljeno 11.07.2017].

8. Kumar L. (2015), *Difference between Compiler and Interpreter* <raspoloživo na: http://techwelkin.com/compiler-vs-interpreter >, [pristupljeno 12.07.2017].

9. Lovrenčić et al. (2009), *50 Years of Higher Order Programming Languages,* <raspoloživo na: http://bib.irb.hr/datoteka/377471.Lovrencic_Konecki_Orehovacki.pdf >, [pristupljeno: 25.08.2017].

10. Lutz M. (2013), *Learning Python – 5th Edition*, O'Reilly Media, <raspoloživo na: http://stock.ethop.org/pdf/python/Learning%20Python,%205th%20Edition.pdf >, [pristupljeno 11.07.2017].

11. Ousterhout J. K. (1998), *Scripting : Higher Level programming for the 21st Century*, <raspoloživo na: http://www.tcl.tk/doc/scripting.html >, [pristupljeno 10.07.2017].

12. Purer K. (2009), *PHP vs. Python vs. Ruby – The web scripting language shootout,* <raspoloživo na: https://klau.si/files/php-vs-python-vs-ruby.pdf >, [pristupljeno 18.08.2017].

13. Schwartz R. L. (2007), *Mastering Perl*, O'Reilly Media, <raspoloživo na: http://index-of.es/Perl/O'Reilly%20-%20Mastering%20Perl%20(2009).pdf >, [pristupljeno 10.07.2017].