UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,

RAČUNALNIŠTVO IN INFORMATIKO

Lidija Jesenek

SPLETNI UČBENIK ZA UČENJE PROGRAMIRANJA Z JEZIKOM JAVASCRIPT

Diplomsko delo

Maribor, marec 2021

UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,

RAČUNALNIŠTVO IN INFORMATIKO

Lidija Jesenek

SPLETNI UČBENIK ZA UČENJE PROGRAMIRANJA Z JEZIKOM JAVASCRIPT

Diplomsko delo

Maribor, marec 2021

SPLETNI UČBENIK ZA UČENJE PROGRAMIRANJA Z JEZIKOM JAVASCRIPT

Diplomsko delo

Študent(ka):	Lidija Jesenek
Študijski program:	Univerzitetni študijski program
	Medijske komunikacije
Smer:	Medijska produkcija
Mentor(ica):	doc. dr. Marko Hölbl, univ. inž. rač. in inf.
Somentor(ica):	doc. dr. Lili Nemec Zlatolas, univ. dipl. med. kom.

ZAHVALA

Zahvaljujem se mentorju Marku Hölblu in somentorici Lili Nemec Zlatolas za usmerjanje pri pisanju diplomskega dela, vse popravke in odzivnost.

Še posebno bi se rada zahvalila družini za podporo skozi vsa leta šolanja in pa Danielu, ki me je dodatno spodbujal.

Spletni učbenik za učenje programiranja z jezikom JavaScript

Ključne besede: JavaScript, Blockly, vizualni programski jeziki, i-učbenik, učenje programiranja

UDK: [37.018.43:004.7]:004.42(043.2)

Povzetek

Z namenom ustvarjanja okolja za učenje programiranja JavaScripta v slovenskem jeziku smo preučili možnost učenja z vizualnimi programskimi jeziki. Pregledali smo že obstoječa okolja na spletu za učenje programskih jezikov. Z ugotovitvami smo sestavili spletni učbenik za učenje osnovnih konceptov programskega jezika JavaScript. S pomočjo knjižnice Blockly in zbirke nalog smo zagotovili, da je spletni učbenik interaktiven in enostaven za uporabo.

A web-based textbook for learning programming in JavaScript

Key Words: JavaScript, Blockly, visual programming languages, interactive textbook, learning programming

UDK: [37.018.43:004.7]:004.42(043.2)

Abstract

With the purpose of creating an environment for learning programming in Javascript, we studied the possibility of using visual programming languages. We made an overview of already existing web platforms for learning programming. With our findings we built a web-based textbook for learning basic JavaScript concepts. With the help of Blockly library and a collection of exercises, we ensured that the textbook is interactive and easy to use.

KAZALO

KAZALO SLIK

KAZALO TABEL

1 UVOD

1.1 Opredelitev problema

Ker so znanstvene in tehnološke inovacije postale ključna sestavina gospodarske rasti, je vse več zanimanja za izobraževanje na teh področjih. Zaradi hitre rasti tehnologij, ki temeljijo na programski opremi ali jo posredujejo, postaja ena izmed ključnih spretnosti kot osnova pismenosti v prihodnosti računalniško programiranje [15].

Istočasno postaja E-učenje vse bolj razširjeno [40]. Ljudem ponuja več možnosti za nadaljevanje izobraževanja, saj je učenje preko spleta dostopno vedno, brez časovnega in prostornega konflikta z delom, družino ali fizičnim obiskovanjem drugih predavanj oziroma šole. E-učenje omogoča učenje na način, ki je za uporabnike smiseln in ugoden. Prav tako je izobraževanje preko spleta in tehnoloških naprav prisotno že v osnovnih šolah, na primer uporaba interaktivnih tabel, uporaba spletnega okolja Moodle, uporaba spleta z interaktivno vsebino kot pripomoček za motivacijo ali uporaba spleta za dodatno razlago snovi.

Prav učenje računalniškega programiranja je ena od dostopnejših tem na spletu. Najdemo lahko veliko spletnih mest z razlago, snovjo in tudi z nalogami. Ponujenih je ogromno brezplačnih ali plačljivih spletnih tečajev za učenje programiranja v različnih programskih jezikih. Da razvijemo dobre programerske sposobnosti od, nas zahteva veliko vaje in vsaj v začetku dodatno razlago. Eden najbolj rabljenih in popularnih jezikov v svetu programiranja ter jezik, s katerim se pogosto srečamo v načrtu učne snovi na programu informatike in sorodnih študijskih smeri, je JavaScript.

V slovenščini je dostopnega e-gradiva za učenje programiranja, v primerjavi z angleškimi viri, relativno malo, zato želimo ustvariti e-učbenik, ki bo celotno vsebino predstavil v slovenskem jeziku. Da bo učbenik kar najbolj sodoben in zanimiv za uporabo, bo tudi interaktiven. Interaktivni učbenik ali i-učbenik (pogovorno še vedno e-učbenik) je e-učbenik, ki ima poleg besedil in grafičnih prikazov dodane multimedijske gradnike, kot so hiperpovezave, videi, zvoki, animacije in interaktivne vsebine, kot so kvizi, samopreverjanje, simulacije itd. [3]. Vključili bomo možnost uporabnikove interakcije z učbenikom z nalogami, ki jih bo moč reševati, nato pa bo

prikazana pravilnost ali nepravilnost naše rešitve. Osredotočili se bomo na določen programski jezik – JavaScript. JavaScript je zaradi svoje sintakse enostaven za začetnike v programiranju.

1.2 Namen diplomskega dela

Namen diplomske naloge je, da se ustvari interaktivni spletni učbenik – interaktivno spletno okolje, kjer se bo uporabnik lahko seznanil s temo učenja in nato pridobljeno znanje prav tako sam preveril.

V spletnem učbeniku bomo predstavili osnovne koncepte programskega jezika JavaScript. Z implementacijo Blocklyja, JavaScriptove knjižnice za ustvarjanje vizualnih blokov, ki se med seboj povezujejo, bo mogoče sestavljati pomenske bloke JavaScript kode in preveriti njihovo pravilnost. Uporabnik se bo tako naučil ne le teoretične vsebine, pač pa bo preko interaktivnega vmesnika svoje znanje istočasno lahko preizkusil preko izbranih nalog in izzivov iz programiranja z JavaScriptom.

Cilj je torej ustvariti delujoč spletni učbenik za učenje JavaScripta v slovenskem jeziku, z možnostjo interakcije uporabnika, kar pomeni, da bo omogočena izmenjava informacij med uporabnikom in učbenikom.

1.3 Predpostavke in omejitve

Pri izdelovanju interaktivnega spletnega učbenika za učenje osnov programiranja se bomo omejili na skriptni programski jezik JavaScript. Vsebina spletnega učbenika bo vsebovala koncepte programiranja z JavaScriptom, saj bo učbenik postavljen za uporabnike brez programerskega predznanja. Predstavljene bodo spremenljivke, polja, pogojni stavki, zanke, funkcije.

Pri pregledu sorodnih tehnologij za učenje programiranja se bomo omejili na spletna prosto dostopna, brezplačna, mesta v slovenskem jeziku, ki spadajo med bolj poznana in popularna.

Predpostavljamo še, da bomo pri pregledu literature o načinih učenja programiranja pretežno uporabljali literaturo v angleškem jeziku, saj je je na to temo na voljo več kot v slovenskem jeziku.

1.4 Predvidene metode dela

V diplomski nalogi bomo uporabili naslednje metode:

- Pregled sorodnih tehnologij za učenje programiranja in učenje jezika JavaScript, ki so primerne za začetnike in so dostopne na spletu, brezplačne, v slovenskem ali angleškem jeziku.
- Študija literature o učenju programiranja.
- Načrtovanje in razvoj spletnega učbenika.

2 UČENJE PROGRAMIRANJA NA SPLETU

Internet in tehnologija sta z ustvarjanjem digitalnih knjižnic, večnamenskih namizij, distribucijo informacij preko elektronske pošte in navsezadnje z uvedbo novega tipa izobraževanja, imenovanega spletno učenje ali e-učenje, spremenila tradicionalno izobraževanje. Najpomembnejše prednosti tega novega tipa izobraževanja so preprost dostop do informacij, fleksibilnost, priročnost, prihranek časa, večopravilnost, povezovanje, različni pristopi. Tako so se razvile spletne platforme za izobraževanje [20].

Okolja za učenje programiranja, kot jih je razdelil Horváth [14]:

- Spletne platforme za učenje v obliki tečajev, kot so Codecademy [6], ali Khan Academy [24]. Te uporabniku predstavijo samo majhne količine novega materiala naenkrat. Za uvajanje nove teme uporabljajo besedilne opise ali pa je tema predstavljena v obliki videa. Za vajo nato vsebujejo spletni urejevalnik s samodejnimi testi. Take platforme bi bilo sicer težko uporabiti kot del učnega načrta v izobraževalnih ustanovah, saj ne vsebujejo vseh običajnih korakov reševanja nalog, kot so načrtovanje, ročno testiranje in razhroščevanje (ang. debugging).
- Spletne tekmovalne platforme, kot so CodeChef [7] ali Codewars [9]. Poleg kompilatorja za kodo vsebujejo še opis nalog, avtomatizirane teste za preverjanje pravilnosti rešitev in včasih tudi manualne teste. Na teh platformah uporabniki tekmujejo med sabo z reševanjem nalog različnih težavnosti oziroma sodelovanjem na tekmovanjih, pri čemer si prislužijo določeno število točk.
- Spletni urejevalniki, ki so osredotočeni na reševanje izobraževalnih izzivov, kot je repl.it [36]. Integriranemu razvojnemu okolju z vgrajeno konzolo so dodana uporabna izobraževalna orodja, kot je vodenje učilnice, ustvarjanje nalog, avtomatizirani testi, spremljanje dejavnosti učencev in dodajanje nalog z opisi.

Avtor H. Alrubaye [19] v svojem delu opredeli dodatno obliko spletnih okolij za učenje programiranja, to so okolja na osnovi blokov, kot so Pencil Code [34], Scratch [37], App

inventor [27]. Takšna okolja predstavljajo programerske ukaze v prijazni obliki in barvi, za lažjo dostopnost, pomnjenje in uporabo.

Drugi viri za učenje, ki jih velja omeniti, so še vsi video krožki, kjer je ves material (brez možnosti reševanja nalog) dostopen naenkrat, na primer tečaj JavaScript fundamentals na Youtube kanalu LearnCode.academy [25], učni viri v obliki dokumentacije, kot je spletno mesto Mozilla Developer Network Web Docs [30], spletna mesta z bazami vaj skupaj z rešitvami, kot je W3Resource [41], izobraževalne video igre, kot je Code Combat [8] in pa interaktivne spletne platforme z možnostjo reševanja nalog, kot sta freeCodeCamp [12] ter W3C Schools [35] [42]. Nabor virov na spletu je torej zares velik in raznovrsten.

Pri vseh prednostih, ki nam jih ponuja spletno učenje programiranja, pa ima takšen način učenja tudi nekaj pomanjkljivosti [1]:

- pomanjkanje interakcije s fizično osebo oziroma omejeno učenje iz oči v oči, ki lahko zmanjša občutek povezanosti, kar lahko vpliva na motivacijo učenca,
- stopnja opustitve takšnega načina učenja je v primerjavi s klasičnim šolanjem večja,
- redke takšne platforme na spletu so pedagoško standardizirane,
- preverjanje prisotnosti učencev pri izpitih je omejeno,
- način učenja preko spleta je primernejši za samostojno usmerjeno učenje, zato se študenti, ki niso vešči samoupravljanja, soočajo s težavami.

3 VIZUALNI PROGRAMSKI JEZIKI

Vizualni programski jezik je programski jezik, ki uporabniku omogoča ustvarjanje programov predvsem skozi grafično manipulacijo [11], namesto na bolj tradicionalen, besedilni način. Vizualizacija služi kot pomoč za razumevanje abstraktne vsebine programskih jezikov. Z vizualnimi programskimi jeziki se vizualno prikaže sintaksa programskih jezikov, večkrat tudi prikažejo vizualno izvedbo programa [32].

Nekateri pogosti modeli vizualnih programskih jezikov so [11]:

- vlečenje blokov po ekranu (na primer Scratch),
- uporaba diagramov poteka, diagramov stanja in drugih komponentnih diagramov (ang. component diagrams) (na primer Pure Data),
- uporaba ikon ali nebesedilnih predstavitev (na primer Kodu).

Vsak vizualni programski jezik ima svojo slovnico in besedišče. Slovnica je vizualna metafora, ki jo jezik uporablja; to so na primer bloki, žice. Besedišče je zbirka ikon, blokov ali drugih komponent, ki nam omogočajo izražanje idej [11].

Pri učenju programiranja začetniki pogosto ne razumejo osnovnih konceptov, kot so spremenljivke, zanke, rekurzija, vhodni in izhodni ukazi in kako so ti izrazi izraženi s programskim jezikom, oziroma se morajo naučiti specifične sintakse, ki jim je najprej nerazumljiva. Vse to vodi v težave s pisanjem algoritmov in pisanjem programov. Po drugi strani lahko opazimo, da učenci, ki se dobro naučijo sintakse jezika in osnovnih konceptov, pogosto ne znajo sestaviti delujočega programa v celoti [32]. V več študijah je bilo potrjeno, da se je programiranja ne le težko naučiti, ampak je programiranje tudi težko učiti [3].

Vizualni jeziki se uporabljajo kot učni pripomoček za uvod v programiranje in koncepte programiranja. Veliko blokovnih vizualnih jezikov uporablja združevanje programskih ukazov s pomočjo oblike in barve za lažjo dostopnost, pomnjenje in uporabo [1]. Ker bloki že vsebujejo ukaze v obliki teksta ali ikon, se učencu ni treba naučiti natančne sintakse, oziroma to učencu pomaga, da se izogne napakam v sintaksi. Prav tako si

učencu ni potrebno zapomniti vseh programskih ukazov. Takšen način učenja je bolj poenostavljen od kompleksnejšega tradicionalnega učenja programiranja, ki se pogosto začne prav z učenjem sintakse. Po drugi strani pa so nekatere raziskave pokazale, da učenci vizualnim elementom vedno ne dajo pomena oziroma vizualiziranih konceptov ne povežejo s programsko prakso [23].

V raziskavi omenjeni v delu Visual and textual programming languages: A systematic review of the literature [29] se je izkazalo, da so učenci, stari od dvanajst do štirinajst let, bili boljši in imeli večjo samoučinkovitost pri učenju programiranja, če so za to uporabljali vizualne bloke, kot če so uporabljali zgolj tekst. Medtem je raziskava na srednješolskih učencih pokazala, da izbira med vizualnim programskim jezikom Alice ali tekstovnim programskim jezikom za učenje samo ni imela večjega pomena, se je pa izkazalo, da so učenci, ki so uporabljali Alice, imeli več volje za nadaljevanje učenja programiranja in računalništva. To bi lahko pripisali večjemu občutku zabave med uporabljanjem vseh vizualnih programskih jezikov na splošno [29].

3.2 Blockly

Blockly je uporabniška (ang. client-side), odprtokodna knjižnica za dodajanje že pripravljenih vizualnih blokov s sintaktično pravilno kodo v aplikacije. Prvič je bil javno objavljen leta 2012. Nastal je kot projekt Googla in je brezplačna programska oprema. Je knjižnica JavaScripta in ima možnost izvoza kode v več programskih jezikov, kot so JavaScript, Python, PHP, Lua, Dart, ustvarijo pa se lahko tudi generatorji za druge besedilne jezike. Knjižnica Blockly se lahko uporabi kot del spletih strani ali pa se vgradi v WebView. Na voljo so tudi izvorne (ang. native) Android in iOS verzije. Poleg sestavljanja vizualnih blokov, Blockly podpira tudi več funkcij za ustvarjanje aplikacij. Tako se lahko uporablja za animiranje likov na zaslonu, ustvarjanje scenarijev, upravljanje robotov, in celo generiranje legalnih dokumentov [11].

Nekaj Blocklyjevih prednosti, ki jih navajajo pri Googlu [18]:

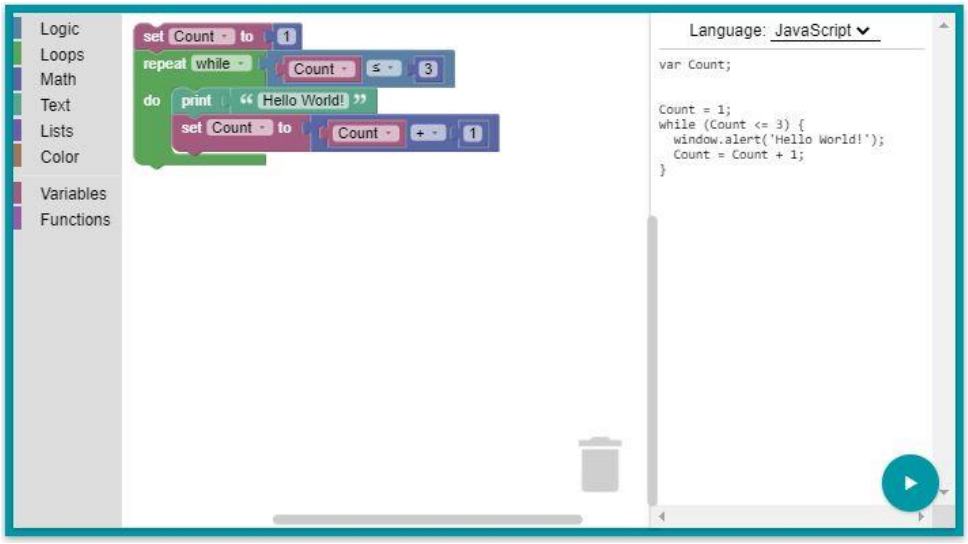
- **Izvozljiva koda.** Uporabniki lahko svoje blokovne programe izvozijo v običajne programske jezike in preidejo na besedilno programiranje.

- **Odprtokodnost.**
- **Razširljivost.** Blockly omogoča dodajanje blokov po meri za APIje ali pa odstranjevanje nepotrebnih blokov in funkcionalnosti.
- **Visoka sposobnost.** Mogoče je izvesti zapletene programerske naloge, kot je izračun standardnega odklona v le enem bloku.
- **Mednarodnost.** Blockly je bil preveden v več kot štirideset jezikov.

Ravno zaradi teh prednosti, z razširljivostjo in visoko sposobnostjo v ospredju, je Blockly v zadnjih letih pridobil na popularnosti pred nekaterimi drugimi okolji na osnovi blokov. V poglavju Primerjava sorodnih del v slovenskem jeziku bomo lahko videli, da je bil Blockly uporabljen pri vseh omenjenih delih, ne le zato, ker je bil preveden v slovenščino. Za večino drugih okolij, ki temeljijo na blokih, namreč velja, da je z njimi težko zgraditi velike, kompleksne programe [19].

Velja opozoriti, da Blocklyja ne dojemamo kot samostojni programski jezik, niti to ni aplikacija, ki je pripravljena za uporabo. Ponuja besedišče in predstavitev za programiranje z bloki, ki jih razvijalec uporabi v svoji aplikaciji. Bloke se lahko ročno povleče preko ekrana in se jih pripne na druge bloke. Ko razvijalci uporabljajo Blockly, sami ustvarjajo svoje blokovne jezike. Pri tem morajo upoštevati stil, katere bloke bodo uporabili, kateri aplikacijski programski vmesnik in katere jezikovne funkcionalnosti bodo uporabili [11].

Blocklyjev uporabniški vmesnik je na osnovni ravni sestavljen iz menija z bloki, ki so razporejeni po skupinah in pa polja, kamor te bloke odlagamo in jih sestavljamo skupaj, po tem, ko smo našli ustrezne v meniju (slika 1). V meniju so bloki razporejeni v skupine, kot so pogoji, zanke, matematične funkcije, tabele, spremenljivke, in funkcije. Za popolno funkcionalnost lahko dodamo še okence z generatorjem, ki vizualne bloke pretvori v besedilno obliko izbranega programskega jezika.



Slika 1: Blocklyjev uporabniški vmesnik [28].

3.3 Primerjava okolij Blockly in Scratch

Programa Blockly in Scratch sta si na prvi pogled zelo podobna; oba za preproste programske naloge uporabljata vizualne bloke, vendar pa je med njima nekaj razlik. Z Blocklyjem uporabnik združuje bloke s tradicionalnimi programskimi stavki. Čeprav ponuja omejeno število vnaprej pripravljenih blokov (in s tem stavkov), uporabniku omogoča ustvarjanje novih blokov po meri, ki bodo zadovoljili njegove potrebe. Uporabnik ima tudi možnost, da bloke s kodo pretvori v besedilne programske jezike.

Scratch je platforma za učenje programiranja, ki je primarno oblikovana in namenjena otrokom. Platforma ponuja okolje, ki vsebuje okolje z razširjeno knjižnico blokov, s katerimi otroci ustvarjajo zgodbe, animacije ali igre. Vsebuje različne vaje in gradi skupnost učencev in učiteljev [26].

Značilnost	Blockly	Scratch
Ustvarjanje blokov po meri	Da	Da
Prevod v besedilne jezike	Da	Ne
Učno okolje prijazno otrokom	Ne	Da
Skupnost članov za deljenje	Ne	Da
Zmogljivost za kompleksne projekte	Da	Ne
Možnost priprave okolja na spletni strani	Da	Ne
Predpripravljeni bloki kode	Omejeno število	Obsežno število

Tabela 1: Primerjava lastnosti Blocklyja in Scratcha [29].

Scratch je torej bolj prijazen otrokom, medtem ko se Blockly bolj uporablja za programiranje orodij, ki se dejansko uporabljajo v resničnem svetu (tabela 1), kot so Code.org, Ozo Blockly micro:bit (ozobot.com), Gameblox.org. Leta 2015 sta se ekipi Google in Scratch sestali, da bi ustvarili novo verzijo Scratcha z uporabo Blocklyjeve tehnologije. Kot rezultat je leta 2019 bila izdana nova verzija Scratch 3.0 [26].

4 JAVASCRIPT

4.1 Zgodovina JavaScripta

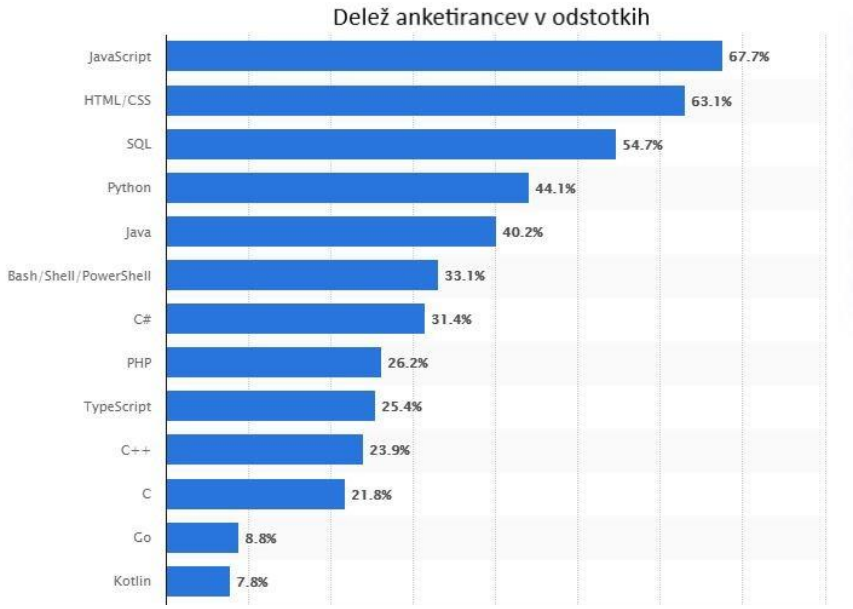
JavaScript je bil ustvarjen zaradi želje po dinamičnih spletnih straneh, zatem, ko se je stran naložila v spletnem brskalniku. Razvijalci Mosaica, prvega spletnega brskalnika (izdan leta 1993) prijaznega za ljudi brez predhodnega tehničnega znanja, so leta 1994 ustanovili podjetje Netscape corporation in izdali nov brskalnik, Netscape Navigator, ki je kmalu postal najbolj razširjen brskalnik. Zaradi potreb uporabnikov in tržnega trenda so ugotovili, da bi splet moral biti bolj dinamičen. Netscape si je za to nalogo zaželel skriptnega jezika, ki bi po sintaksi bil podoben Javi, ki je takrat bil eden popularnejših jezikov. Leta 1995 je torej Brendan Eich za Netscape napisal prototip, ki so ga najprej poimenovali Mocha, potem LiveScript, kmalu pa je bil preimenovan v JavaScript.

K dogajanju se je priključila ECMA (European Computer Manufacturers Association – Evropsko združenje proizvajalcev računalnikov), ki je definirala standard ECMAScript. Medtem je Microsoft po zgledu Navigatorja razvil svoj bralec JScript za brskalnik Internet Explorer, s katerim je Netscape izgubil bitko za popularnost. Zaradi popularnosti tega brskalnika je bil v standard ECMA vključen JScript (in ne JavaScript).

Ko je leta 2005 Jessee James Garrett izdal vodnik o setu tehnologij, ki so temeljili na Javascriptu in ob tem prvič uporabil pojem Ajax, se je zanimanje za JavaScript spet povečalo. To je spodbudilo rast skupnosti in odprtokodnih knjižnic kot jQuery, Prototype, Dojo Toolkit in MooTools. Google je leta 2008 izdal svoj brskalnik Chrome z V8 JavaScript motorjem, ki je bil hitrejši od konkurence. Zahvaljujoč temu je bil konec leta 2008 izdan ECMAScript 5 standard, ki je končno vseboval JavaScript.

4.2 Popularnost jezika Javascript v zadnjih letih

Glede na raziskavo, narejeno za potrebe spletnega mesta statista.com, je bil JavaScript z letom 2020 najbolj uporabljen programski jezik. Skoraj oseminšestdeset odstotkov anketirancev je navedlo, da uporabljajo JavaScript. Sledijo mu HTML/CSS, SQL, Python in Java (slika 3).



Slika 2: Najbolj uporabljeni jeziki razvijalcev po svetu z začetkom leta 2020 [39].

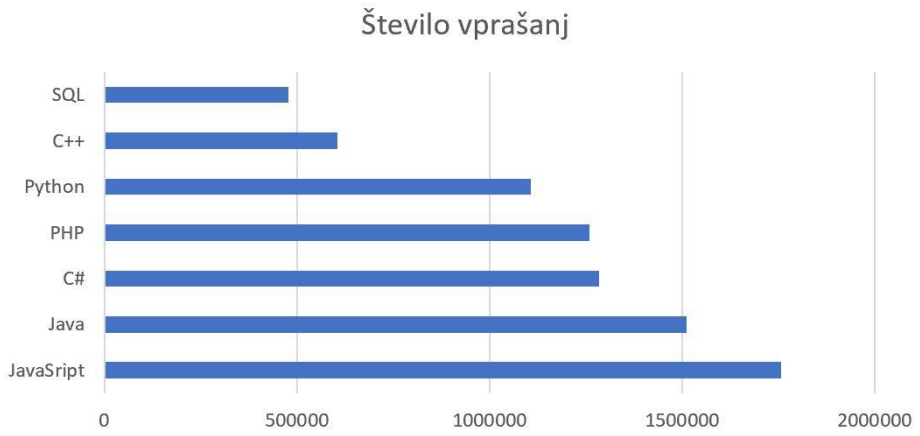
JavaScript je vsesplošno uporabljen v vseh spletnih brskalnikih, vendar pa je bilo z letom 2009, ko je izšlo okolje Node.js, prvič možno ustvariti aplikacije z JavaScriptom na strani strežnika. Tudi zaradi tega je JavaScript univerzalno uporabno orodje. Vredno je omeniti, da na popularnosti pridobiva TypeScript, ki je razširjena verzija JavaScripta.

Prednosti JavaScripta, kot jih navaja Ballard, so [33]:

- odpiranje novih oken s specifično velikostjo, položajem in stilom,
- omogoča uporabniku prijazno pomoč, kot je spustni meni,
- preverjanje podatkov, vnešenih v spletne obrazce - da se zagotovi, da so podatki v ustrezni obliki (formatu), preden je obrazec poslan spletnemu strežniku,
- spreminjanje, kako elementi spletne strani izgledajo in se obnašajo, ko se zgodi določen dogodek, kot premikanje miške nad njimi,
- odkrivanje in izkoriščanje naprednih funkcij, podprtih z uporabo določenega brskalnika, kot so tretjeosebni vtičniki, ali domorodna podpora novim tehnologijam.

4.3 Prednosti in slabosti učenja JavaScripta kot prvega programskega jezika

Učenje JavaScripta kot prvega programskega jezika ima veliko prednosti. Za začetek učenja je potrebno zelo malo; da bi zagnali kodo JavaScripta, potrebujemo le spletni brskalnik. Ni nam potrebno namestiti prevajalnika kode in nastaviti kompleksnega okolja. To poenostavi tudi deljenje kode. Drugi zelo pomembni dejavnik je to, da ima JavaScript zelo veliko skupnost. Glede na poročilo Global App Testing o najbolj povpraševanih jezikih na spletni strani Stack Overflow je JavaScript jezik, o katerem se na splošno največ sprašuje [31]. Število vprašanj za jezik JavaScript je v letu 2019 presegalo milijon in pol. Več kot milijon in pol vprašanj je bilo zadanih le še za jezik Java, sledita C# in PHP z okoli milijon in tristo tisoč vprašanji, Python z nekoliko več kot milijon vprašanji, za jezik C++ je bilo vprašanih nekaj več kot pol milijona vprašanj, za vse ostale jezike pa manj kot pol milijona (slika 3).



Slika 3: Graf števila vprašanj na StackOverflow za nekatere bolj popularne programske jezike [31].

To pomeni, da uporabnik JavaScripta z lahkoto najde sredstva, kadar ima težave pri programiranju, saj lahko na strani, kot je Stack Overflow vpraša za pomoč in dobil bo odgovore, iz katerih se lahko uči.

JavaScript je vsesplošno uporaben; ima veliko zbirko orodij, ki jih je mogoče uporabiti v zelo različnih aplikacijah, od mobilnih aplikacij do vizualizacije podatkov.

Obstajajo tudi slabosti učenja JavaScripta kot prvega programskega jezika. Razredi v JavaScriptu so drugačni od standardov v drugih programskih jezikih, zato spoznavanje objektno orientiranega programiranja v JavaScriptu ni priporočljivo. Večkrat se poudarja, da to ni najboljši programski jezik za spoznavanje globlje vede o računalništvu. Veliko stvari v JavaScriptu je v primerjavi z drugimi jeziki storjenih avtomatično, zato se lahko zgodi, da oseba, ki se uči le JavaScripta, ne bo poznala pojmov, kot so podatkovne strukture ali dodeljevanje pomnilnika.

4.4 Zgradba JavaScripta

4.4.1 Spremenljivke in vrednosti

Spremenljivke v JavaScriptu lahko napovemo na več načinov. Pred letom 2015 je bila beseda `var` edini način. Z letom 2015 je bila izdana verzija JavaScript (ES6), ki omogoča uporabo besede `const` za definiranje spremenljivk, ki jih ne moremo spreminjati in pa besedo `let` za definiranje spremenljivk z omejenim obsegom [35]. Spremenljivke uporabljamo, da lahko v njih shranjujemo vrednosti.

Vrednosti so podatki v programu, ki jih v obliki bitov shranjujemo v delovni pomnilnik, da jih lahko kasneje uporabimo. Vsaka vrednost ima svoj tip in vsak tip vrednosti ima v programu drugačno vlogo.

Da v jeziku JavaScript ustvarimo vrednost, zadostuje, da jo pokličemo po imenu. Tukaj se JavaScript razlikuje od »zahtevnejših« programskih jezikov, kjer moramo vrednost najprej definirati sami in jim določiti tip. V Javascriptu se tip določi samodejno. Paziti moramo tudi na to, da JavaScript razlikuje med uporabo velikih in malih črk. Osnovni tipi vrednosti v javaScriptu so:

- **Števila.** Števila so primarni tip spremenljivke v JavaScriptu. Ta tip se uporablja za predstavitev celih števil in približnih vrednosti celih števil.
- **Posebna Števila.** V JavaScriptu poznamo zapis za števila, ki pa to vendar niso. To sta ''infinity'' in ''-infinity''.

- **Nizi.** Nizi se uporabljajo za predstavitev besedila. Zapišemo jih znotraj enojnih ali pa dvojnih narekovajev.
- **Boolean Vrednosti**. Uporabljamo jih za označevanje vrednosti, kjer imamo na razpolago dve možnosti, kot na primer ''Da'' in ''Ne'', ''True'' in ''False'' ali ''On'' in ''Off''.
- **Prazne vrednosti.** V JavaScriptu poznamo še posebne vrednosti, imenovane prazne vrednosti. To sta ''null'' in pa ''undefined''. Sami po sebi so to vrednosti, vendar pa ne nosita nikakršne informacije.

4.4.2 Funkcije

Funkcije uporabljamo kot sredstvo za razstavljanje kode v module za večkratno uporabo. Ko smo funkcijo napisali, je ta na voljo programu, da jo lahko uporablja, kot bi funkcija sama bila del jezika JavaScript [33]. Z uporabo funkcij je program lažje vzdrževati in razhroščiti.

Osnovna sintaksa funkcije zgleda tako:

```
function sayHello() {
        alert (''Hello'')
}
```

Najprej zapišemo besedo function, ki ji sledi ime funkcije po naši izbiri z dodanimi oklepaji in nato še zaviti oklepaji, kamor vstavimo JavaScript stavke. Ime funkcije vedno zapišemo z malo začetnico.

Ko definiramo funkcijo, se ta še ne izvede, za to jo moramo najprej poklicati. Funkcijo pokličemo tako, da preprosto uporabimo ime funkcije z oklepaji, kjer koli želimo, da se izvedejo stavki iz funkcije:

```
sayHello();
```

4.4.3 Polja

Polja so urejena zbirka vrednosti. Vsaka vrednost v polju se imenuje element in vsak element ima svoj oštevilčeni položaj v polju, ki se imenuje indeks [10]. Polja so

uporabna, saj lahko v njih shranjujemo velike količine vrednosti, ki jih pozneje lahko spreminjamo; do posameznih elementov lahko dostopamo tako, da jim spremenimo vrednost ali jih izbrišemo. S pomočjo indeksov lahko elemente dodajamo. Polja lahko tudi urejamo, lahko jih na primer razporedimo po velikosti.

Vsako polje v JavaScriptu ima lastnost `length`, ki nam pove, koliko elementov se nahaja znotraj polja. Ta lastnost se samodejno spremeni, ko izbrišemo elemente v polju ali jih dodamo. Length oziroma dolžina polja je vedno za 1 večja, kot je število elementov v polju, saj je prvi element oštevilčen z indeksom nič.

Ustvarjanje polja je kot ustvarjanje novega objekta, vendar pa za ta podatkovni tip obstaja bližnjica – polje najlažje ustvarimo s pomočjo oglatih oklepajev ([]). Če želimo, lahko polju dodamo elemente istočasno, kot ga ustvarimo, ali pa jih dodamo po tem, ko je polje že bilo ustvarjeno.

```
var myArray = [];
```

Nekatere uporabnejše metode za polja so: concat(), join(), toString(), indexOf(), lastIndexOf(), slice(), sort(), splice().

4.4.4 Pogojni stavki in Zanke

Pogojni stavek je stavek, s katerim lahko izvedemo del kode pod nekim pogojem ali pa naredimo kaj drugega, če ta pogoj ni izpolnjen [22].

Pogojni stavki in zanke v JavaScriptu [5]:

- Stavek **Switch** primerja izraz s številnimi možnimi vrednostmi.
- Stavek **Break** se uporablja za prekinitev stavka switch(). Lahko se tudi uporabi za prekinitev zanke.
- Stavek **Continue** prekine eno iteracijo v zanki, ko se zgodi določen pogoj, in nadaljuje z naslednjo iteracijo v zanki.
- Stavek **If** določa blok kode, ki naj se izvede, če je pogoj resničen.
- Stavek **Else** določa blok kode, ki naj se izvede, če je pogoj neresničen.
- Stavek **Else if** poda nov pogoj, če je prvi pogoj napačen.

- Zanka For je splošna konstrukcija za iteriranje (ang. iterating) elementov. Klasična zanka **For** deluje s števcem ali podobno spremenljivko, ki se posodablja po vsaki ponovitvi.
- Zanka **For of** iterira elemente ponavljajočega objekta, najpogosteje tabele ali niza.
- Zanka **For in** iterira lastnosti objekta.
- Zanka **While** izvrši stavek, medtem ko je pogoj izpolnjen.
- **Do while** iterira blok kode le enkrat in nato ponovi zanko, medtem ko je določen pogoj resničen.

4.4.5 Objekti

Objekti so podatkovne strukture v JavaScriptu, s katerimi je možno doseči veliko fleksibilnosti v naši kodi [13]. V njih lahko shranjujemo bolj ali manj vse: spremenljivke, vrednosti, tabele in celo funkcije. To nam omogoča veliko možnosti. V JavaScriptu obstaja več načinov za ustvarjanje objektov:

- Z **Object literals**. Ta način je zelo preprost, vse kar moramo storiti je, določimo ime, ki bo predstavljalo objekt, in temu dodelimo zavite oklepaje (prazne ali pa z lastnostmi).
- Z besedo **New**. Besedo New uporabimo pred konstruktorjem funkcije. Primer:

```
var emptyObject = new Object();  //Ustvari prazen objekt.
var date = new Date(); //Ustvari object z danim podatkovnim
tipom.
```

- Z **Object.create()**. Kot nam pove ime, Object.create() ustvari nov objekt. Kar ta način razlikuje od prejšnjih je, da uporabi obstoječ objekt kot prototip za novo ustvarjeni objekt. Če na primer definiramo objekt avto tako:

```
var avto = {
   name: '',
   printName: function() {
     console.log('Ime avta je: ' + this.name)
```

```
    }
};
```

Potem lahko uporabimo ta obstoječi objekt za prototip novega:

```
var novAvto = Object.create(avto);
```

Ko je bil objekt ustvarjen, lahko do njega dostopamo in nastavljamo ter spreminjamo njegove lastnosti. To lahko storimo na dva načina, in sicer s piko in z oglatimi oklepaji:

```
var avtoName = avto.name;
var avtoName = avto['ime'];
```

Za nastavljanje ali spreminjanje lastnosti zadostuje, da obstoječo vrednost preprosto prepišemo ali pa vrednost samo nastavimo, če ta še ne obstaja, na primer:

```
avto.name = 'Volvo';
```

Če bi obstoječe lastnosti objekta želeli zbrisati uporabimo kljično besedo ‘delete’ pred imenom lastnosti objekta:

```
delete avto.name;
```

4.4.6 DOM

Document object model (v nadaljevanju DOM) je podatkovna predstavitev objektov, ki sestavljajo vsebino in strukturo dokumenta (strani) na spletu. Vsak tak dokument je sestavljen iz strukturiranih vozlišč, ki so med seboj povezana v odnosu starš-otrok. Zahvaljujoč tej strukturi lahko s skriptnim jezikom dostopamo do elementov znotraj dokumenta in se pomikamo med njimi, kar nam omogoča preoblikovanje teh s številnimi metodami. Elemente v DOMu lahko dodajamo, izbrišemo ali jih spreminjamo.

DOM nam omogoči, da vemo, kje se je določen dogodek zgodil, oziroma da se je zgodil na specifičnem mestu. Predstavlja stran na način, da se v programu lahko spremeni

zgradba, stil in vsebina. DOM predstavi dokument z objekti in `nodes()` in zato se lahko programski jezik poveže s stranjo.

DOM je torej objektno-orientirana predstavitev spletne strani, ki se jo lahko spreminja s skriptnim jezikom, kot je JavaScript. Zgradbo dokumenta predstavlja z odnosom starš-otrok, ki jo lahko uporabljamo za premikanje med elementi.

Najosnovnejša operacija, ki jo moramo pri tem poznati, je izbiranje vozlišč. Metode izbiranja elementov z JavaScriptom in jQueryjem so naslednje:

- izbiranje vozlišč z identifikatorjem ID:
  ```
  var element = document.getElementById(id);
  ```
- izbiranje vozlišč z imenom oznake (tag name):
  ```
  var elements = document.getElementsByTagName(name);
  ```

Tako getElementByID() kot getElementByTagName() sprejmeta le en argument. Razlika je v tem, da kot argument namesto identifikatorja prenesemo ime oznake. GetElementByTagName() nam omogoča, da ustvarimo polje z vsemi ponovitvami določene oznake.

- Izbiranje vozlišč z imenom razreda:
  ```
  var elements = document.getElementsByClassName(names);
  ```

2.4.7 Dogodki

Dogodki so stvari, ki se zgodijo HTML elementom na strani, JavaScript pa se na te dogodke odzove.

Te stvari sproži uporabnik ali pa brskalnik sam, na primer: klik z miško, premik miške, klik na tipkovnico, nalaganje strani, nalaganje dokumentov.

Najpopularnejši dogodki so: onchange, ko se HTML element spremeni; onclick, ko uporabnik klikne HTML element; onmouseover, ko uporabnik premakne miško preko HTML elementa; onmouseout, ko se miška premakne stran od HTML elementa, onkeydown; ko uporabnik pritisne tipko na tipkovnici; onload, ko brskalnik konča nalaganje strani.

V grobem dejanja lahko ločimo na dogodke vnašanja (ang. input events), Mouse events (dogodke miške), Click events (dogodki klikanja), Load events (dogodke nalaganja) in druge dogodke.

Ko JavaScript reagira na določen dogodek, poimenujemo to upravljanje z dogodki (ang. event handling). Ti se uporabljajo za urejanje in potrjevanje uporabniških vnosov, uporabniških dejanj in dejanj brskalnika, kot so na primer:

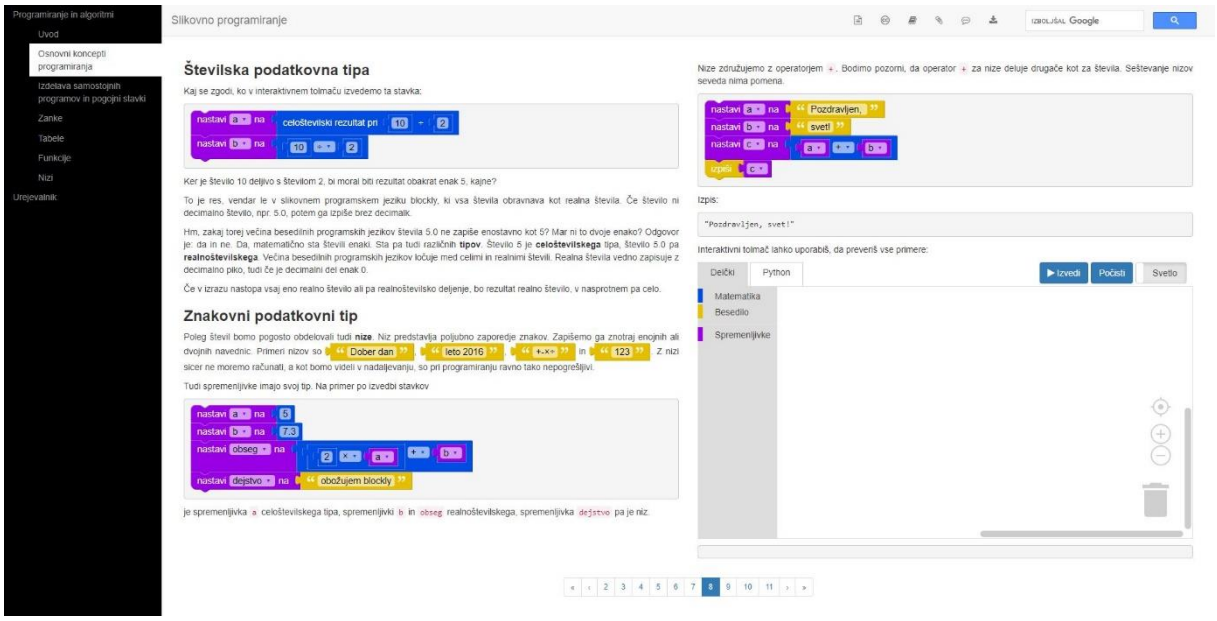
- Stvari, ki se morajo zgoditi vedno, ko se naloži spletna stran.
- Stvari, ki se morajo zgoditi vedno, ko se stran zapre.
- Dejanje, ki se mora zgoditi, ko uporabnik pritisne tipko.

Dogodke se po navadi uporablja istočasno s funkcijo, ki se ne izvrši, dokler se ne zgodi dogodek. Ko se dogodek izvrši, se sproži JavaScript funkcija, ki določi, kaj točno se zgodi po tem, ko se sproži dogodek.

5 PREGLED SORODNIH DEL V SLOVENSKEM JEZIKU

V predpripravi za izdelavo našega izdelka smo preverili, kakšna podobna dela že obstajajo. Ker je bil naš namen ustvariti e-učbenik v slovenščini, smo se tudi pri pregledu omejili le na dela v slovenskem jeziku.

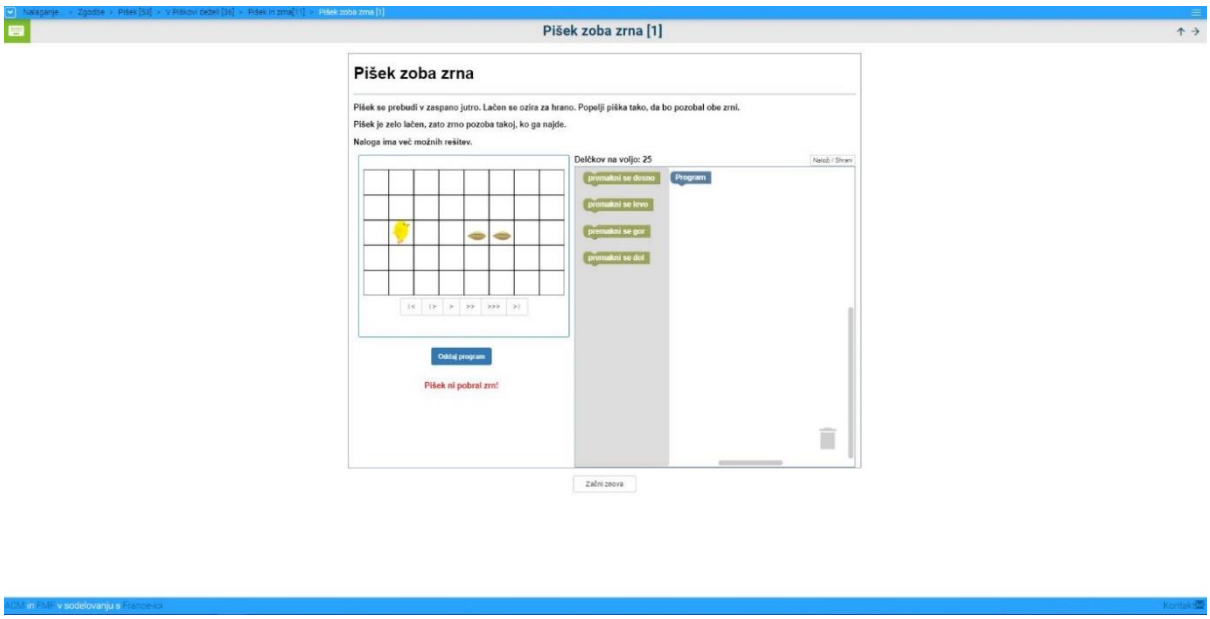
Slikovno programiranje je E-učbenik za uvod v programiranje. Izdan je bil na Fakulteti za računalništvo in informatiko v Ljubljani leta 2018 in je namenjen tako osnovnošolcem in srednješolcem kot tudi učiteljem, z glavnim namenom spodbujanja računalniške pismenosti med slovenskimi najstniki. Programiranje uči s slikovnim programskim jezikom Blockly. Spletni učbenik je brezplačen in prosto dostopen na spletu [13].



Slika 4: Primer strani v e-učbeniku Slikovno programiranje [13].

Uporabnik se v sedmih glavnih poglavjih z devetinosemdesetimi stranmi nauči ukaze in gradnike programskega jezika Python. Vključena je tudi možnost prikaza slikovne kode v tem besedilnem programskem jeziku.

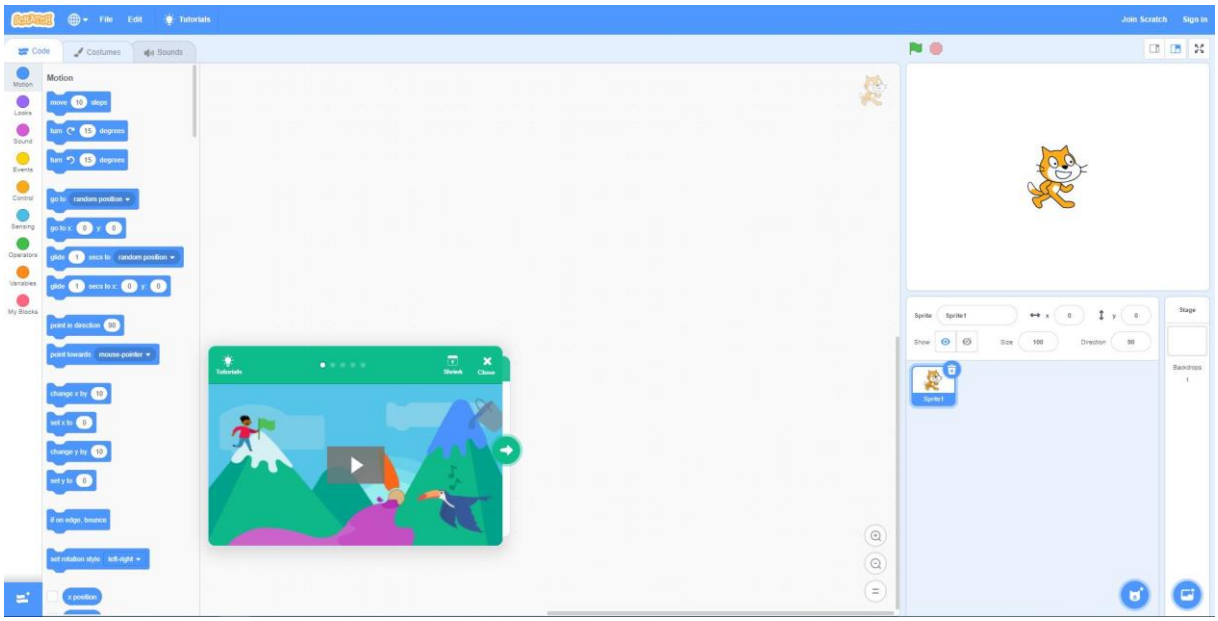
Portal Pišek je spletno mesto, ki je nastalo leta 2018 in bilo razširjeno leto kasneje v sklopu projekta Naloge za poučevanje in učenje računalniškega mišljenja Portal Pišek (NPUR), pri katerem sodelujeta Fakulteta za matematiko in fizikoter Kreativni center Poligon. Prvotni namen tega portala je ponuditi slovenskemu šolskemu prostoru sklop nalog v slovenskem jeziku in povezati naloge v Mednarodnem tekmovanju iz računalniškega razmišljanja – Bober in Srednješolskem tekmovanju ACM iz računalništva in informatike – RTK. Portal temelji na francoskem sistemu Algoréa in reševanju nalog s slikovnim programskim jezikom Blockly [28].



Slika 5: Primer zgodbe na portalu Pišek [28].

Na Pišku so ustvarjene štiri različne metode za pridobivanje znanja programiranja v obliki štirih učnih poti. V učni poti Zgodbe je vključenih osem različnih avtorskih sklopov nalog in več kot dvesto petdeset nalog za reševanje, kjer uporabnik z Blocklyjem ustvari interaktivno zgodbo. V sklopu Programski koncepti najdemo naloge, kjer se uporabnik sreča z dejanskimi programskimi koncepti in ukazi. V sklop Učbenik so bile prenesene naloge iz E-učbenika Slikovno programiranje. Temu sklopu so bile dodane vaje, ki temeljijo na teh nalogah ali pa služijo preverjanju osvojenega znanja. Učna pot Code Week vsebuje izbrane naloge iz prejšnjih sklopov in je namenjen šolam, ki sodelujejo v Evropskem tednu programiranja (EU Code Week) [37].

Scratch je spletna platforma za učenje programiranja z vizualnim programskim jezikom Scratch, ki je na voljo v več kot šestdesetih jezikih, med njimi tudi v slovenščini. Razvit in oblikovan je bil z neprofitno organizacijo Scratch Foundation ter prvič objavljen leta 2007. Zasnovan je predvsem za otroke, stare od osem do šestnajst let. V letu 2019 je projekte s Scratchom ustvarilo več kot dvajset milijonov mladih [37].



Slika 6: Primer strani na platformi Scratch [37].

Scratch je tudi skupnost, saj ima uporabnik možnost deljenja svojih projektov. To so najpogosteje zgodbe, igre in animacije.

Pri pregledu sorodnih del smo ugotovili, da se za slovenska spletna okolja za učenje programiranja uporablja pristop učenja z vizualnimi jeziki. Dve od del uporabljata Blockly, medtem ko tretje delo temelji na Scratchu. V dveh pregledanih delih od treh se okolje za učenje ne osredotoča na noben specifičen programski jezik, pač pa predstavlja okolje za učenje programskih konceptov na splošno, skozi igro. V delu, kjer je omogočeno učenje določenega programskega jezika, to je jezika Python (tabela 2), je način podajanja razlage in nalog v obliki učbenika in ne igre. Spoznali smo, da slovenski prostor ne ponuja dovolj rešitev za učenje programskega jezika JavaScript.

Tudi mi smo se odločili za uporabo metode učenja z vizualnim jezikom Blockly zaradi njegovih prednosti.

	Slikovno programiranje	Pišek	Scratch
Način podajanja znanja	Učbenik	Igra in učbenik	Igra
Uporabljen vizualni jezik	Blockly	Blockly	Scratch
Učenje določenega programskega jezika	Da, Python	Ne	Ne

Tabela 2: Primerjava sorodnih del.

6 IZDELAVA SPLETNEGA UČBENIKA

6.1 Oblikovanje strukture e-učbenika

Da bi lahko določili potrebne tehnologije za razvoj e-učbenika na spletu in nadaljnje oblikovanje uvodnega dela e-učbenika s kodo ter dodajanje vsebine, je bilo potrebno najprej načrtovati osnovno postavitev strani oziroma učnih enot. Pri tem smo se ravnali po usmeritvah v delu slovenski i-učbeniki [17]:

- Primarna ciljna skupina naprav, na katerih se uporabljajo i-učbeniki, so osebni računalniki in tablice; vsebine se predvidoma nahajajo lokalno na napravi, zato daljši čas dostopanja do vsebin ni priporočljiv; mora biti možnost spreminjanja vsebin. Te usmeritve smo rešili z uporabo ustreznih tehnologij pri razvijanju spletne rešitve.
- I-učbenik naj vsebuje kazalo, učni načrt in/ali učbenik, zbirko vaj, zbirko aktivnosti, generator testov; omogočeno mora biti preprosto "listanje" po straneh naprej/nazaj; prehodi med vsebinskimi sklopi morajo biti jasni; če je način uporabe ležeč, naj bo vsebina v dveh stolpcih, razen v primeru, ko je interaktivni gradnik preširok za širino enega stolpca. Te usmeritve smo upoštevali tako, da smo na vsako stran e-učbenika dodali meni, ki je preprost za uporabo. Vse vsebinske sklope smo ustrezno poimenovali. Vsak sklop vsebuje naloge, ki jih je s klikom na gumb »Rešitev« mogoče preveriti. Razlago snovi smo predstavili v dveh stolpcih.
- E-učna enota vsebuje uvod, jedro, povzetek in naloge in najmanj en interaktivni element; e-učna enota je sestavljena iz posameznih strani; vsaka ima svoje ime, ki se pojavi v kazalu na levi strani in v predogledu strani kot glava strani; tip strani dosledno označimo. V našem delu je vsaka stran (učna enota) poimenovana v kazalu in ima v levem zgornjem kotu naslov. Vsaka e-učna enota vsebuje uvod (definicijo koncepta), jedro z označenimi primeri, in pa glavni interaktivni element – Blockly.

6.2 Uporabljene tehnologije in pomembnejši deli izvirne kode

Za izdelavo spletnega učbenika smo uporabili osnovna programska orodja, to so HTML, CSS, ogrodje Foundation in JavaScript s knjižnico jQuery. Implementirali smo Blockly. Razvijali smo v okolju Visual Studio Code.

Struktura aplikacije je preprosta, vsaka podstran ima ločeno datoteko HTML.

Da bi dosegli bolj modularno strukturo in vključili isti stranski meni na vse podstrani, smo uporabili metodo `fetch`.

```
fetch("./side-menu.html")
  .then(response => {
     return response.text()
  })
  .then(data => {
   document.querySelector("side-menu").innerHTML = data;
  });
```

Ta metoda nam je omogočila, da z dodajanjem oznake (ang. tag) v kodo html naložimo datoteko side-menu.html.

```
<side-menu
   </side-menu>
```

Da bi zagnali Blockly smo morali najprej ustvariti objekt Blockly. Navodila za implementacijo Blocklyja smo našli na Googlovi spletni strani [39].

```
var workspace = Blockly.inject('blocklyDiv',
    {toolbox: document.getElementById('toolbox'),
    zoom:
       {controls: true,
        wheel: true,
        startScale: 1.0,
        maxScale: 3,
        minScale: 0.3,
        scaleSpeed: 1.2},
   trashcan: true
});
```

Instanci Blocklyja lahko dodamo dogodek na naslednji način:

```
workspace.addChangeListener(myUpdateFunction);
```

Funkcija dodana k temu dogodku Change se sproži vsakič, ko uporabnik naredi kakšno spremembo. Zahvaljujoč tej možnosti smo lahko bloke interaktivno spremenili v kodo.

Da bi kodo v besedilni obliki lahko sedaj pokazali uporabniku, smo uporabili metodo za vstavljanje kode, ki jo generira knjižnica Blockly.

```
function myUpdateFunction(event) {
  code = Blockly.JavaScript.workspaceToCode(workspace);
  $('#code-area').html(PR.prettyPrintOne(code));

}
```

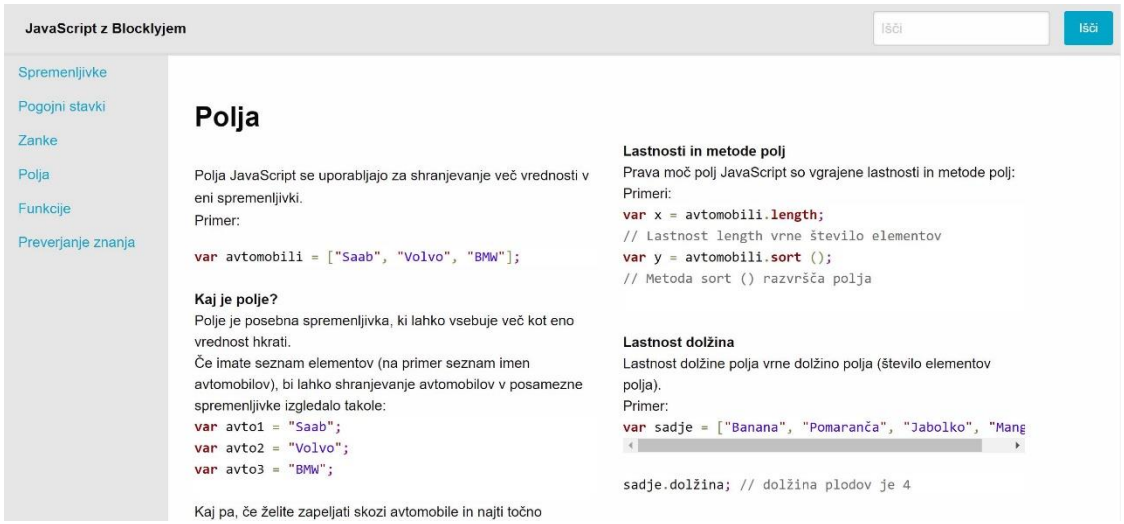
Potem smo za prednji del naredili gumb 'Zaženi'. S klikom nanj je poklicana funkcija `executingCode()`, ki s pomočjo vgrajene funkcije eval() zažene prikaz besedilne kode v JavaScriptu:

```
<input value="Zaženi" class="button" type="button"
id="codeToString" onClick="executingCode();">
```

6.3 Sestava in vsebina e-učbenika

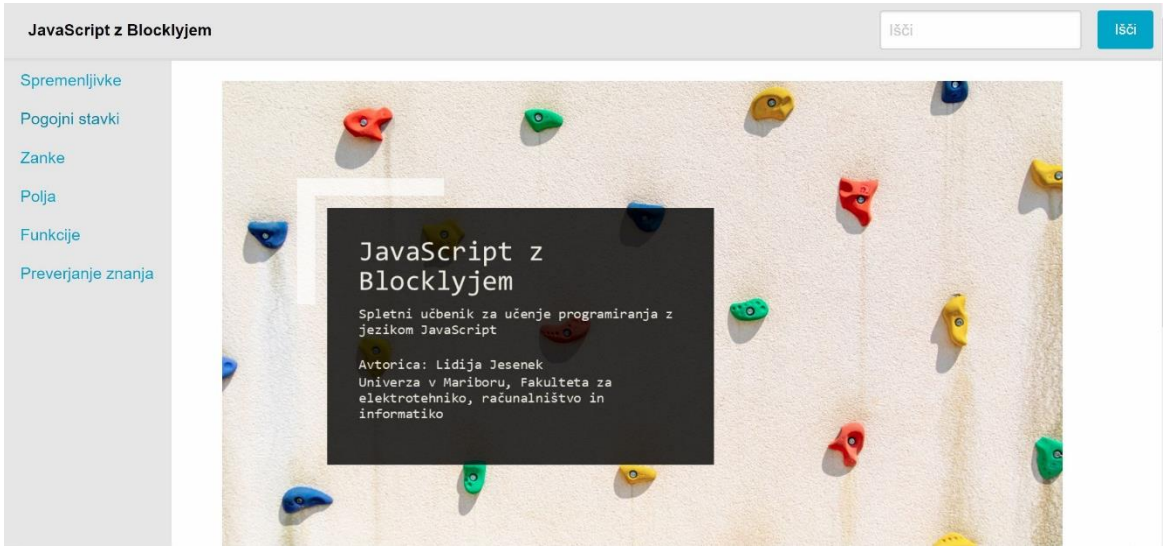
Zbrali smo ustrezno literaturo z opredelitvijo konceptov programiranja in s pomočjo tega na e-učbenik vključili podajanje razlage. Ker je večina virov, ki so nam bili na voljo, v angleškem jeziku, smo morali angleške izraze prevesti v slovenščino. Pri tem so nam bile v pomoč spletne strani, kot so DIS slovarček [4], Termania [38] in Islovar [16].

Po študiju možnosti, ki jih koncepti v JavaScriptu ponujajo, in pregledom sorodnih spletnih platform za učenje programskih jezikov z nalogami, smo zbrali nabor nalog, ki smo jih vključili v ustrezne sklope. Vsebina e-učbenika je sestavljena iz petih večjih sklopov, ki predstavljajo najosnovnejše koncepte v JavaScriptu, to so Spremenljivke, Funkcije, Pogojni stavki, Zanke in Polja (slika 7). Dodali smo še šesti sklop Preverjanje znanja, ki je namenjen uporabniku za samopreverjanje znanja, pridobljenega skozi celoten učbenik.



Slika 7: Primer strani v spletnem učbeniku.

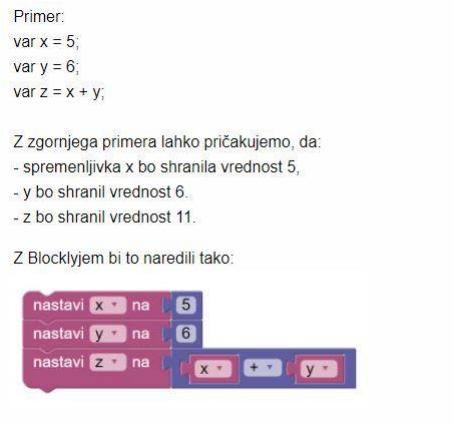
Vrstni red sklopov (slika 8) smo določili glede na sorodna dela, literaturo in lastni občutek pri izkušnjah z učenjem programskih jezikov. Najprej smo predstavili spremenljivke, ki so najosnovnejši koncept, brez katerega bi težko uporabljali nadaljnje koncepte. Potem smo uvedli pogojne stavke, s katerimi lahko nadzorujemo potek programa. Spoznali smo zanke, s katerimi lahko upravljamo z več podatki. Takoj za zankami smo predstavili polja, ki se večkrat uporabljajo skupaj z zankami za iteracijo elementov polj. Nazadnje smo predstavili funkcije, v katere lahko vključimo vse omenjene koncepte in jih nato v programu večkrat uporabimo.



Slika 8: Naslovna stran z menijem.

V spletni učbenik nismo vključili koncepta objektov, saj ti niso podprti s strani Blocklyja. Kot lahko vidimo v uradnem repozitoriju knjižnice [16], so podprte naslednje skupine blokov: barve, polja, logika, zanke, matematika, procedure, napisi, spremenljivke, dinamične spremenljivke. Dodajanje takšne funkcionalnosti bi bilo zelo zahtevno in bi zahtevalo ogromno vrstic kode.

Vsak sklop vsebuje razlago pojmov in naloge, s katerimi lahko učenec oziroma študent preveri svoje znanje. V razlago smo vključili tudi primere, ki smo jih za lažje reševanje nalog v nadaljevanju prikazali tudi s sestavljenimi bloki (slika 9).

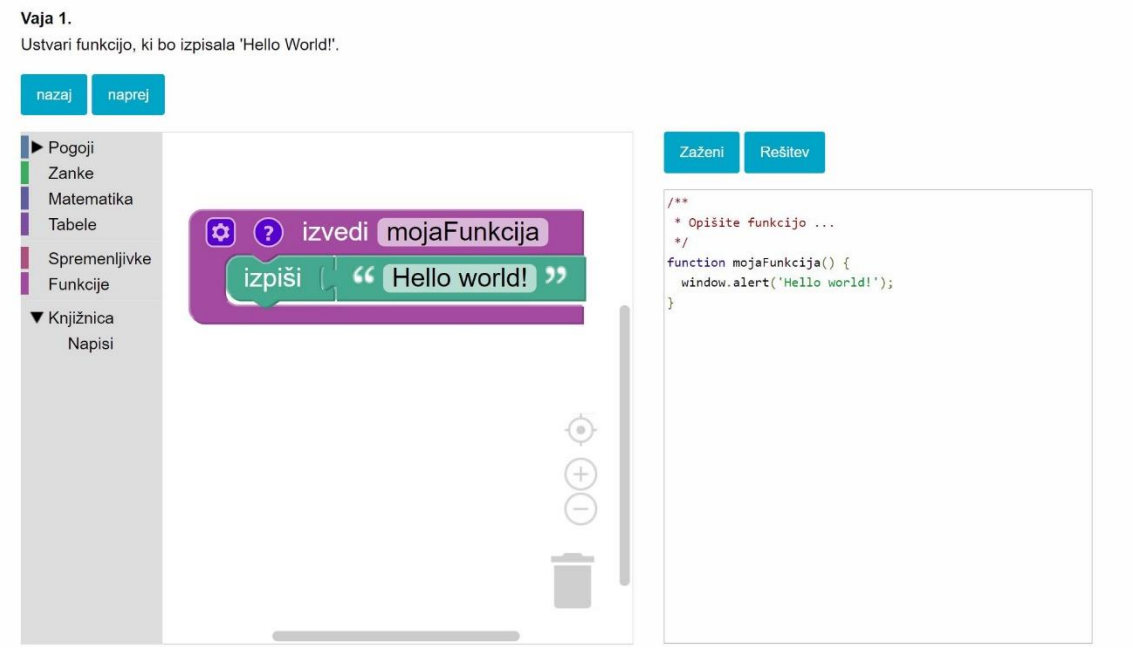


Slika 9: Primer razlage z bloki Blocklyja.

Razlagi sledijo vaje. Vsak sklop vsebuje nekaj vaj, ki se prikazujejo s pomikanjem s pomočjo gumbov naprej in nazaj (slika 10).

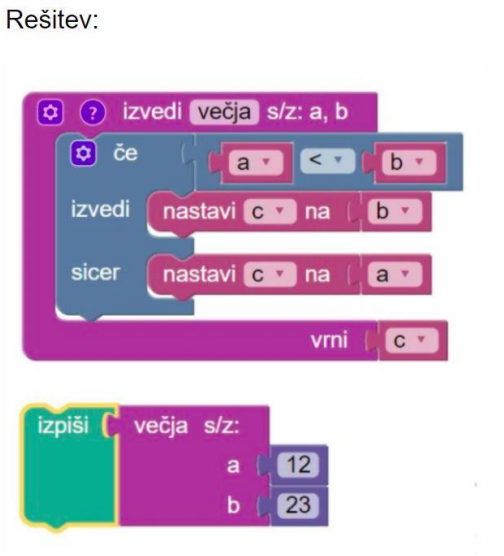
Slika 10: Primer naloge.

Uporabnik oziroma učenec programiranja preveri svoje znanje s sestavljanjem blokov v zato namenjenem okencu Blocklyja. V okencu poleg tega se postavljeni bloki sočasno pretvarjajo v jezik JavaScript (slika 11).



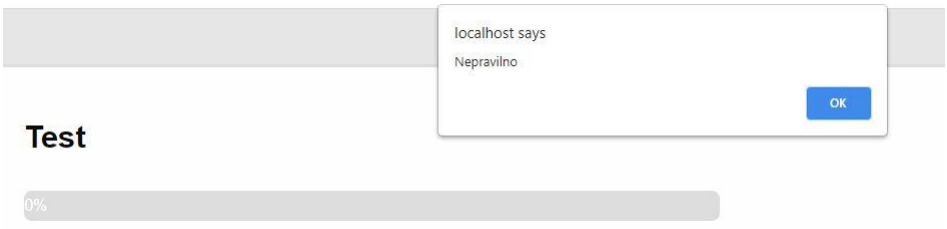
Slika 11: Reševanje nalog.

Uporabnik spletnega učenika lahko preveri pravilnost rešene naloge s klikom na gumb Rešitev. Rešitev se nato prikaže pod rešeno nalogo, v obliki sestavljenih blokov (slika 12). Za vse naloge, ki smo jih vključili v spletni učbenik smo torej preverili, ali so z Blocklyjem sploh rešljive. Pri nobeni od nalog nismo naleteli na težavo, da reševanje z Blocklyjem ne bi bilo mogoče.



Slika 12: Prikaz rešitve.

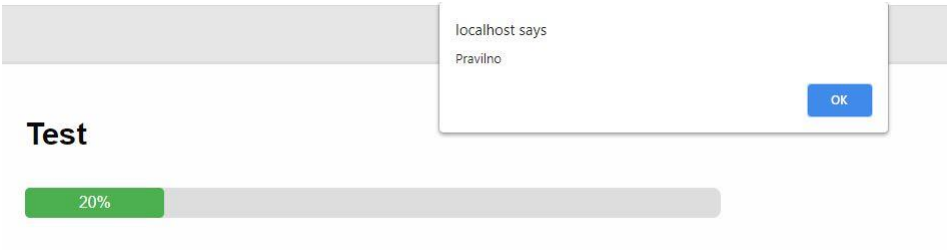
V zadnjem sklopu spletnega učbenika, imenovanem Preverjanje znanja lahko uporabnik preveri, če je stopnja njegovega pridobljenega znanja zadovoljiva. V ta sklop smo vključili pet nalog, izmed katerih vsaka povzema enega od prejšnjih sklopov. Te naloge se od nalog, vključenih v prejšnje sklope, razlikujejo po načinu preverjanja pravilnosti rešitve. Vsaka naloga vsebuje funkcionalnost za testiranje pravilnosti rešitve, ki preveri, če je izhod (ang. output) napisanega programa pravilen. Funkcija `eval`, ki izvaja kodo, vrne vrednost programa, ki ga je napisal uporabnik. To vrednost nato primerjamo z vrednostjo, ki smo jo v kodi definirali kot pravilno. Uporabnik pravilnost svoje rešitve v sklopu Preverjanje znanja preverja s pritiskom na gumb Preveri. Če po kliku na ta gumb program ugotovi, da se vrednosti ne ujemata, bo brskalnik uporabniku izpisal besedo »Nepravilno« (slika 13), če pa se vrednosti ujemata, se bo izpisala beseda »Pravilno!«.



Slika 13: Izpis brskalnika, ko se vrednosti ne ujemata.

Če ima uporabnik težave z reševanjem nalog, lahko rešitev znova preveri s klikom na gumb Rešitev. Tokrat se rešitev prikaže v obliki kode JavaScript in ne v obliki sestavljenih blokov Blockly, kot smo bili navajeni v prejšnjih sklopih. Od uporabnika pričakujemo, da zna brati kodo v besedilni obliki. Uporabnik lahko vidi, kolikšen del testa je izpolnil pravilno s pogledom na vrstico napredka (ang. Progres bar), ki smo jo naredili s pomočjo CSSa in JavaScripta. Napredek uporabnika se prikazuje v zeleni barvi. Privzeto stanje vrstice napredka je 0% (slika 13). Vsakič, ko uporabnik v sklopu Preverjanje znanja pravilno reši nalogo, se poveča dolžina vrstice napredka v zeleni barvi (slika 14). Prirastek dolžine smo naredili s pomočjo JavaScripta, tako da izberemo element in njegovo lastnost `width` povečamo na višjo propocionalno vrednost:

```
width = Number(width) + Number(20);
    elem.style.width = width + "%";
    elem.innerHTML = width  + "%";
```



Slika 14: Procent se z vsako pravilno rešeno nalogo poveča.

7 ZAKLJUČEK

V diplomskem delu smo proučili obstoječe spletne portale za učenje programiranja. V nadaljevanju smo preučili možnosti vizualnih (slikovnih) programskih jezikov in ugotovili, da so takšni jeziki primerni za začetne korake v programiranju. Skozi zgodovino in značilnosti JavaScripta smo razumeli, zakaj je ta jezik eden najpopularnejših za učenje in uporabo.

Preučili smo že obstoječa spletna brezplačna mesta za učenje programiranja v slovenskem jeziku. Glede na njihove pomanjkljivosti smo se lotili izdelave lastnega e-učbenika za učenje programiranja v jeziku JavaScript. Po usmeritvah za pripravo spletnih učbenikov, ki smo jih našli v literaturi, smo oblikovali zgradbo spletne strani. Za izdelavo smo uporabili osnovna orodja; e-učbenik je oblikovan s pomočjo HTML, CSS in ogrodja Foundation ter z JavaScriptom s knjižnico jQuery za večjo interaktivnost. V e-učbenik smo vključili okence, kamor uporabnik dodaja vizualne bloke iz knjižnice Blockly. Pri sestavljanju blokov uporabnik sliši zvočni učinek. Ustvarjeno kodo v sliki lahko uporabnik zmanjša, poveča, postavi v sredino okenca ali pa bloke izbriše. Temu smo dodali generator besedilnega programskega jezika JavaScript, ki se prikazuje v okencu zraven. Implementirali smo razlago osnovnih konceptov v JavaScriptu po sklopih (spremenljivke, pogojni stavki, zanke, polja in funkcije) skupaj s primeri za boljše razumevanje. Razlagi vsakega koncepta smo dodali naloge za reševanje z Blocklyjem in rešitve teh nalog za samopreverjanje. Dodali smo zakjučni sklop Preverjanje znanja, kjer lahko uporabnik preveri svoje znanje, pridobljeno skozi vsa poglavja učbenika. Za dodatno motivacijo pri reševanju testa smo vključili vrstico napredka, ki zabeleži pravilno rešene naloge. Z vkjučitvijo razlage snovi, primerov in vaj v slovenskem jeziku v naš e-učbenik smo dosegli, da se lahko vsak slovensko govoreči človek nauči osnov programskega jezika JavaScript.

Ugotovili smo, da je programski jezik JavaScript dober jezik za začetek učenja programiranja, saj na spletu obstaja nešteto virov (v angleškem jeziku) za učenje tega jezika in ima ta jezik na spletu veliko skupnost. Osvojili smo osnovno znanje o tem jeziku in to znanje nato uporabili, ko smo razvili spletno stran. Ugotovili smo tudi, da so

Blockly in ostali vizualni programski jeziki primerni za začetek učenja programiranja, po drugi strani pa imajo omejene možnosti uporabe. V našem e-učbeniku prav zaradi tega ni možno narediti z Blocklyjem vsega, kar se z JavaScriptom da narediti v resničnem svetu, na primer uporabljati objekte.

Spletni učbenik bi lahko še razširili ter ga testirali na uporabnikih. Vključili bi lahko možnost učenja in reševanja nalog kompleksnejših lastnosti JavaScripta, kot so dogodki, objekti in knjižnica jQuery. Da bi dosegli večje zanimanje za učenje in učinkovitost pri učenju, bi lahko povečali interaktivnost učbenika z uporabo elementov igre, kot je na primer zbiranje točk, ali pa bi vključili možnost komentiranja in s tem grajenje skupnosti. Preučevanje JavaScripta bi lahko nadaljevali z bolj kompleksnimi koncepti. Lahko bi tudi nadaljevali z preučevanjem vseh možnosti, ki jih ponujajo drugi vizualni jeziki.

8 VIRI IN LITERATURA

[1] A. Alaqsam and F. M. Ghabban, “Online Programming Language Learning Using Massive Open Online Courses in Saudi Universities 1,” vol. 9, no. 2, 2021.

[2] A. Berglund and R. Lister, “Introductory Programming and the Didactic Triangle,” Conferences in Research and Practice in Information Technology Series, vol. 103, no. Ace, pp. 35–44, 2010.

[3] A. Čuk, I. Pesek, and B. Zmazek, “Slovenski i-učbeniki,” 2014, [Online]. Available: http://www.zrss.si/pdf/slovenski-i-ucbeniki.pdf.

[4] Amebis, “Slovarski portal Termania,” 2020. https://www.termania.net/ (accessed Mar. 15, 2021).

[5] C. S. Horstmann, Modern JavaScript for the Impatient. Addison-Wesley Professional, 2020.

[6] Codecademy, “Codecademy,” 2021. https://www.codecademy.com/ (accessed Mar. 18, 2021).

[7] CodeChef, “CodeChef,” 2021. https://www.codechef.com/ (accessed Mar. 18, 2021).

[8] CodeCombat Inc., “Code Combat,” 2021. https://codecombat.com/ (accessed Mar. 19, 2021).

[9] Codewars, “Codewars,” 2021. https://www.codechef.com/ (accessed Mar. 18, 2021).

[10] D. Flanagan, JavaScript: The Definitive Guide, 7th Edition. O’Reilly Media, Inc., 2020.

[11] E. Pasternak, R. Fenichel, and A. N. Marshall, “Tips for creating a block language with blockly,” Proceedings - 2017 IEEE Blocks and Beyond Workshop, B and B 2017, vol. 2017-November

[12] freeCodeCamp, “freeCodeCamp,” 2021. https://www.freecodecamp.org (accessed Mar. 19, 2021).

[13] G. Anželj, J. Brank, A. Brodnik, L. Fürst, and M. Lokar, “Slikovno programiranje: E-učbenik za uvod v programiranje.” Založba Fakultete za računalništvo in informatiko, Univerza v Ljubljani, Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, 2018, [Online]. Available: https://lusy.fri.uni-lj.si/ucbenik/prog/index.html.

[14] G. Horváth, “A web-based programming environment for introductory programming courses in higher education,” Annales Mathematicae et Informaticae, vol. 48, pp. 23–32, 2018.

[15] G. J. Contreras and K. W. M. Siu, “Computer Programming for All: A Case-Study in Product Design Education,” Procedia - Social and Behavioral Sciences, vol. 182, pp. 388–394, 2015,

[16] Google Blockly on GitHub, 2020. https://github.com/google/blockly/tree/master/blocks (accessed Aug. 26, 2020).

[17] Google developers, “Injection Options,” 2021. https://developers.google.com/blockly/guides/configure/web/configuration_struct (accessed May 18, 2020).

[18] Google Developers, “Introduction to Blockly,” 2020. https://developers.google.com/blockly/guides/overview.

[19] H. Alrubaye, “Comparison of visual programming and hybrid programming environments in transferring programming skills R · I · T,” 2017.

[20] I. Cetina, D. Goldbach, and N. Manea, “Udemy: A case study in online education and training,” vol. 3, pp. 46–54, 2018.

[21] Inštitut “Jožef Štefan,” “DIS slovarček,” 2018. http://dis-slovarcek.ijs.si/ (accessed Mar. 15, 2021).

[22] J. Pollock, JavaScript: A Beginner’s Guide, 5th ed. McGraw-Hill Education, 2019.

[23] J. Sorva, J. Lonnberg, and L. Malmi, “Students’ ways of experiencing visual program simulation,” Computer Science Education, vol. 23, no. 3, 2013.

[24] K. Academy, “Khan Academy,” 2021. www.khanacademy.org (accessed Mar. 18, 2021).

[25] LearnCode.academy, “Javascript fundamentals,” 2015. https://www.youtube.com/watch?v=fGdd9qNwQdQ&list=PLoYCgNOIyGACTDHuZtn0qoBdpzV9c327V&ab_channel=LearnCode.academy (accessed Mar. 19, 2021).

[26] M. Gavin, “Blockly vs Scratch: A detailed comparison,” 2021. https://codingnemo.com/blockly-vs-scratch/#Should_you_choose_Blockly_or_Scratch.

[27] M. I. of Technology, “App Inventor,” 2021. https://appinventor.mit.edu/ (accessed Mar. 19, 2021).

[28] M. Lokar, G. Jerše, and K. K. Oljšak, “O Pišku,” 2019. http://pisek.acm.si/contents/4903/ (accessed Nov. 12, 2020).

[29] M. Noone and A. Mooney, “Visual and textual programming languages: A systematic review of the literature,” arXiv, no. December, 2017, doi: 10.1007/s40692-018-0101-5.

[30] Mozilla, “Mozilla Developer Network Web Docs,” 2021. https://developer.mozilla.org/en-US/ (accessed Mar. 19, 2021).

[31] N. Roberts, “Picking Apart Stack Overflow; What Bugs Developers The Most?,” 2019. https://www.globalapptesting.com/blog/picking-apart-stackoverflow-what-bugs-developers-the-most.

[32] O. Iskrenovic-Momcilovic, “Choice of visual programming language for learning programming,” International Journal of Computers, vol. 2, pp. 250–254, 2017.

[33] P. Ballard, JavaScript in 24 Hours, Sams Teach Yourself, Sixth Edit. Sams, 2015.

[34] Pencil Code, “Pencil Code,” 2021. https://pencilcode.net/ (accessed Mar. 19, 2021).

[35] Refsnes Data, “W3C Schools,” 2021. https://www.w3schools.com (accessed Mar. 19, 2021).

[36] Replit Inc., “repl.it,” 2021. https://replit.com/ (accessed Mar. 18, 2021).

[37] Scratch Foundation, “O Scratchu,” 2021. https://scratch.mit.edu/about (accessed Jan. 15, 2021).

[38] Slovensko društvo Informatika, “Islovar,” 2020. http://www.islovar.org/islovar (accessed Mar. 15, 2021).

[39] statista.com, “Most used programming languages among developers worldwide, as of early 2020,” 2020. https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/ (accessed Sep. 20, 2020).

[40] Š. Hošková-Mayerová and Z. Rosická, “E-Learning Pros and Cons: Active Learning Culture?,” Procedia - Social and Behavioral Sciences, vol. 191, pp. 958–962, 2015, doi: 10.1016/j.sbspro.2015.04.702.

[41] W3resource, “W3Resource,” 2021. https://www.w3resource.com (accessed Mar. 19, 2021).

[42] 42 Ż. Jażdżyk, “Duża paczka materiałów do nauki JavaScriptu,” 2017. https://www.nettecode.com/duza-paczka-materialow-javascript/.