Dizajniranje didaktičkih igara pomoću HTML5 i JavaScript-a

Vlaho, Domagoj

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:886785

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies - INFORI Repository

zir.nsk.hr

DIGITALNA KNJIŽNICA

Image not found or type unknown

Sveučilište u Rijeci – Odjel za informatiku

Informacijski i komunikacijski sustavi

Domagoj Vlaho

Dizajniranje didaktičkih igara pomoću HTML5 i JavaScript-a

Diplomski rad

Mentor: prof. dr.sc. Nataša Hoić-Božić

Rijeka, prosinac 2018.

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 12.3.2018.

Zadatak za diplomski rad

Pristupnik: Domagoj Vlaho

Naziv diplomskog rada: Dizajniranje didaktičkih igara pomoću HTML5 i JavaScript-a

Naziv diplomskog rada na eng. jeziku: Using HTML5 and JavaScript for Serious Game Design

Sadržaj zadatka: Kod učenika mlađe dobi od 1. do 4. razreda osnovne škole koji se prvi puta susreću sa informatikom i kreću u savladavanje elemenata programiranja i algoritamskog razmišljanja, neophodno je razvijanje logičkog i algoritamskog razmišljanja koje im omogućuju rješavanje problema pomoću računala. Navedeni elementi informatike nerijetko su učenicima teško razumljivi i nedovoljno zanimljivi prilikom učenja. Jedno od rješenja ovog problema je prikaz gradiva na njima zabavan i bliži način korištenjem didaktičkih igara odnosno poučavanja temeljenog na igrama (GBL, tj. Game Based Learning). Među današnjim tehnologijama za izradu ovakvih igara su i one temeljene na opće prihvaćenom HTML 5.0 standardu. Glavni element HTML-a koji omogućuje rad sa 2D objektima, te je time prikladan za izradu igara je <canvas>. Osim HTML i CSS kao neophodan dio standarda za razvoj igara je i JavaScript. Zadatak diplomskog rada je izrada odabranih didaktičkih igara za razvoj algoritamskog razmišljanja i učenje koncepata programiranja kod učenika mlađe dobi korištenjem HTML i JavaScript tehnologija.

Mentor:

Prof. dr. sc. Nataša Hoić-Božić

Voditelj za diplomske radove:

Dr. sc. Martina Ašenbrener Katić

Zadatak preuzet: 22.3.2018.

(potpis pristupnika)

Sadržaj

1. Uvod 1
2. GBL (Game-Based-Learning) 2
3. Tehnologije modernih web stranica 3
4. „Trojstvo“ web-a 5
   4.1. HTML 5
   4.2. CSS 7
5. JavaScript 8
   5.1 Javascript kod unutar web preglednika 9
6. Načini uključivanja Javascipt kôda u web projekt 13
   6.1. Interno uključivanje 13
   6.2. Uključivanje eksterne Javascript datoteke 14
7. Javascript sintaksa 15
   7.1. Stil pisanja 16
8. Bootstrap 16
   8.1. Responzivni web dizajn 17
   8.2. Bootstrap grid layout 19
9. Opis izrađene interaktivne stranice s igrama 20
   9.1. Početna stranica 22
   9.2. Informacijska stranica 23
   9.3. Stranica s uputama za igre 25
   9.4. Stranica za selekciju igre (“meni” za odabir igre) 27
   9.5. Tetris stranica 29
   9.6. Stranica odabira memory igre 35
   9.7. Memory igra 37
   9.8. Labirint igra 45
10. Zaključak 50
11. Popis slika 51
12. Literatura 53

1. Uvod

Ovaj diplomski rad vezan je uz područje GBL (*engl. Game-Based-Learning*), odnosno, cilj mu je izrada didaktičkih web igara koje će se koristiti kod učenika mlađe dobi (od 1. do 4. razreda). Razlog uporabe GBL modela u edukacijskim svrhama je taj što djeca (pogotovo djeca mlađeg uzrasta) ponajbolje uče i savladavaju gradivo kroz igru. Sve igre bit će dizajnirane ka tome da potiču razvijanje logičkog i algoritamskog razmišljanja, koje je neophodno kod savladavanja osnovnih informatičkih koncepata i elemenata programiranja.[1]

Za implementaciju web igara, koriste se HTML5 i JavaScript (jezik koji se koristi na webu za ostvarivanje interaktivnosti na stranicama) tehnologije. HTML (HyperText Markup Language) je jezik označavanja koji se koristi za strukturiranje i predstavljanje sadržaja na internetu. On je temelj svake stranice, te se u ovom radu koristi za definiranje osnovnog „kostura“ stranice koja će sadržavati edukativni materijal (didaktičke igre). HTML5 kao novija verzija (revizija osnovnog HTML standarda) donosi nove elemente, bitne za upravljanje multimedijskim sadržajem. Neki od njih su <video>, <audio>, <canvas>, od kojih se u ovom radu primarno koristi <canvas> element, jer omogućuje crtanje i manipulaciju 2D objekata. <canvas> element je svojevrsni grafički „kontejner“ , te za crtanje unutar njega i ostvarivanje interaktivnosti na stranici se koristi JavaScript programski jezik.

Cilj ovog rada je izrada odabranih didaktičkih igara za razvoj algoritamskog razmišljanja i učenje koncepata programiranja kod učenika mlađe dobi. Rad započinje teorijskim opisom GBL modela učenja, zajedno sa opisom osnovnih web tehnologija (HTML, CSS i Javascript) gdje je naglasak stavljen na Javascript komponentu. Prethodno opisana implementacija web igara, detaljno se opisuje nakon teorije. Kod opisa igara je također naglasak stavljen na opis Javascipt datoteka i načina rada unutar istih.

2. GBL (Game-Based-Learning)

GBL je način učenja kroz igru, tj. igranje koja ima definirane ishode učenja. Ovim načinom, uravnotežuje se sadržaj učenja sa igrom kroz koju se sadržaj uči. [2]

Glavni razlog stvaranja GBL načina učenja su djeca i njihov način razmišljanja. Oni provode većinu svog vremena kroz igru, bilo to od igre skrivača, do igranja video igara. Može se reći da su za njih igra i učenje sinonimi, koji dovode do kognitivnog i emocionalnog razvoja unutar društvenog i kulturnog konteksta. [1] Tj., GBL možemo podijeliti na dvije komponente: igru i predmet učenja koji je sadržan u njoj. Igraća komponenta ovakvog načina rada, zaokupljuje djecu, te ona uče efektivnije od tradicionalnog načina učenja. Djeca zajedno sa učiteljima rade kroz igru prema određenom cilju. Odabiru akcije (točne/netočne u skladu s predmetom ispitivanja), doživljavaju posljedice tih akcija [3], tj., kroz eksperimentiranje aktivno uče i vježbaju. Ovakvim aktivnim načinom rada, gradivo se usvaja efektivno, te se jednostavno prenosi iz simuliranog okruženja u stvarni život.

Ukratko, glavne prednosti za korištenje GBL pristupa, možemo svesti u 4 točke:

- Integracija učenja i igre čini učenje zabavnim
- Motivacija učenika za rad
- Kroz igru se djeca zaokupljuju i uče efektivnije
- Potiče se da uče od vlastitih grešaka

Sve web igre ovog rada, bit će rađene u skladu sa GBL pristupom. Kroz igru, koja je djeci najbliža, razvijat će se računalno, tj. algoritamsko razmišljanje koje je potrebno kod računalnog programiranja s kojim će se djeca sresti u višim razredima osnovne škole. Odnosno, igre će biti namijenjene kao uvod djece u koncepte računalnog razmišljanja i programiranja općenito, te će se koristiti kao dodatni aktivni način rada u nastavi.

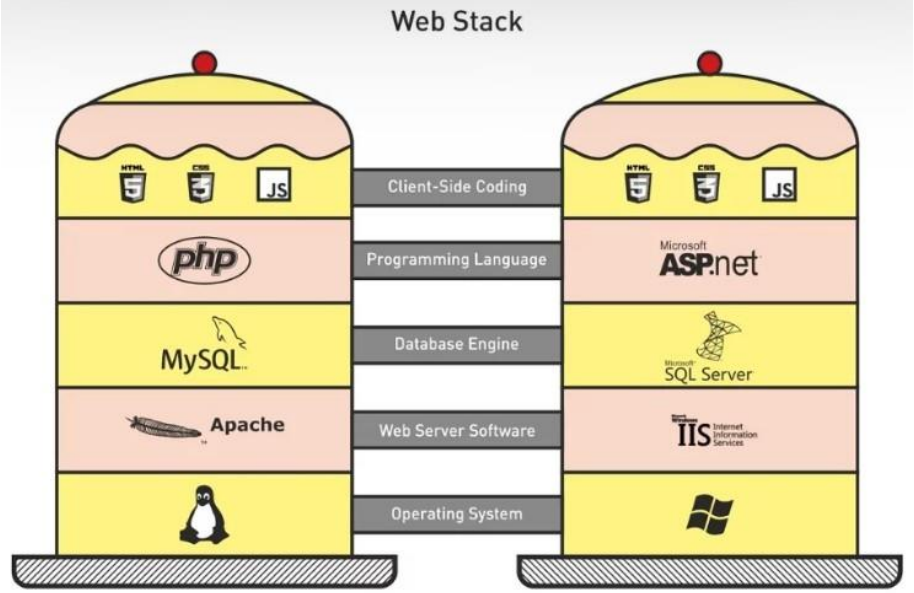


Slika 1 - Game Based Learning ilustracija

3. Tehnologije modernih web stranica

Temelj svake web stranice su tri tehnologije web razvoja (HTML, JavaScript, CSS - Eng. *Cascading-Style-Sheets*) koje je grupnim nazivom moguće nazvati „Klijentske tehnologije“.

Web stranice treba promatrati kao cjelinu različitih tehnologija, gdje svaka radi određeni dio i svaka ima točno određenu funkciju. Odnosno, web stranicu treba gledati kroz slojevitu strukturu (npr. Kao tortu): [4]

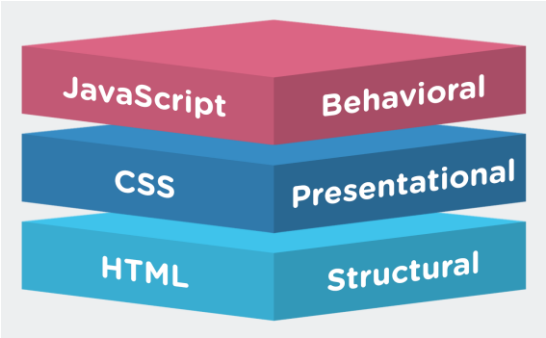


Slika 2 - Slojevita struktura Web-a

Iz *Slika 2 - Slojevita struktura Web-a* je vidljiva spomenuta slojevita struktura moderne web stranice. Na vrhu je sloj „Klijentskih tehnologija“. Naziv klijentski je zbog toga što se izvođenje ovog kôda vrši na klijentskom računalu. Njegova osnova su tri tehnologije (HTML, JavaScript i CSS), te su one temelj svake web stranice. Bez njih nije moguće napraviti web stranicu. Svaka od tih tehnologija koristi se za određeni dio stranice.

Osnovna razlika (a i međusobna povezanost) između ove tri tehnologije je u kratkom opisu definicije i svrhe svake od njih: [5]

- HTML je temelj web stranice i koristi se za kreiranje osnovne strukture web stranice i umetanja sadržaja na istu
- CSS se koristi za definiciju dizajna web stranice (kako stranica izgleda kad ju korisnik otvori)
- JavaScript se koristi za ostvarivanje interaktivnosti i kreiranje interaktivnih elemenata koje zaokupljuju korisnika u sadržaj stranice



Slika 3 - Osnovna "trojka" svake web stranice

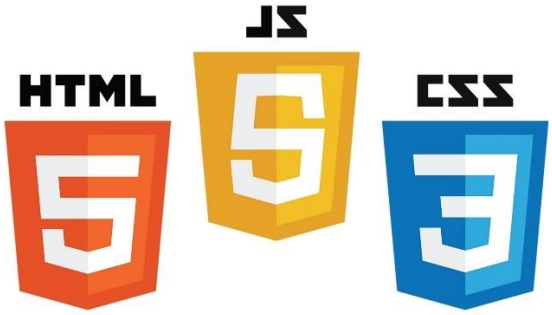
Ostatak tehnoloških slojeva ispod Klijentskih tehnologija su „Serverske tehnologije“ jer se njihovo izvođenje vrši na serveru s ciljem ostvarivanja poslovne logike ili dodatne funkcionalnosti web stranice.

Prema *Slika 3 - Osnovna "trojka" svake web stranice*, prvi od njih je sloj nazvan „Programming language“. Postoje mnogi, no najčešći su: PHP(*PHP: Hypertext predprocesor – serverski skriptni jezik dizajniran primarno za web razvoj*), ASP .NET, Java i Ruby. Idući sloj („Database engine“) ili web server/baza podataka je mjesto gdje se spremaju svi podaci sa određene web stranice. Neke od tih informacija su npr. proizvodi, narudžbe, transakcije, korisnički podaci. Postoji mogućnost spremanja i sadržaja web sjedišta uz pomoć CMS(Eng. *Content-Management-Systems)*. Izbor odgovarajuće platforme baze podataka ovisi o serverskom programskom jeziku, samoj implementaciji servera i još mnogo drugih faktora. Najčešće platforme baze podataka modernih web stranica su MySQL, Microsoft SQL Server, Oracle i Postgres. „Web server software“ sloj u ovom slučaju promatramo kroz softversku stranu servera (bez obzira što je bitna i hardverska implementacija servera). Server općenito dostavlja web stranicu korisniku, a odabir njegovog softvera primarno ovisi o kombinaciji tehnologija koje koristimo u ostalim slojevima web sjedišta. Dva najpoznatija serverska softvera su Apache (Linux) i ISS (Eng. *Internet-Information-Services,* Microsoft) . Zadnji sloj („Operating Systems“) ili operacijski sustav, radi na serverskom hardveru i osigurava da svi slojevi web sjedišta rade kako treba. Ovisno o serverskom softveru, dva najkorištenija sustava su Linux i Microsoft Windows. [4]

Međutim, razvoj modernih web stranica ne možemo promatrati samo kroz navedene tri klijentske tehnologije. Istina, one su temelj svakog front-end developera, no danas postoje i druge tehnologije nadodane na osnovne tri (React, Angular, Vue, SASS - Eng. Syntactically-awesome-style-sheets, …). One ubrzavaju kodiranje, osigurava se veća odvojenost i čitljivost kôda, što je bitno u profesionalnom okruženju. Ovaj rad dotiče se samo prvog sloja prethodno opisane slojevite strukture web stranica. Razlog tome je što sama web stranica neće biti povezana na bilo kakav server ili bazu podataka nad kojima će vršiti spremanje, obradu bilo kakvih podataka. Stoga, glavne tehnologije koje su korištene su HTML i JavaScript, uz dodatak CSS-a.

4. „Trojstvo“ web-a

Trojstvo web-a su tri osnovne tehnologije svake web stranice: HTML, CSS, JavaScript. Prva od njih i osnova svega je HTML.



Slika 4 - HTML, CSS i JavaScript logotipi

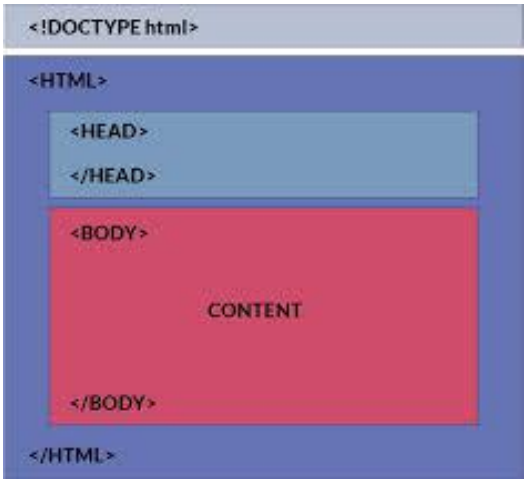
4.1. HTML

HTML je ljepilo koje drži svaku web stranicu. Kao i kod gradnje kuća, za svaku je potreban jak temelj. HTML je temelj web-a. On je jezik otvorenog kôda, kojeg je lako naučiti i za njegovo savladavanje nije potreban nikakav specijalizirani softver. Dovoljan je tekst editor (npr. *Windows Notepad*) i internetski preglednik s pomoću kojeg se testira kod i prikazuje web stranica. [6]

HTML prezentacijskim jezikom oblikuje se sadržaj i stvaraju se hiperveze hipertekst (tekstualna struktura koja se sastoji od međusobno povezanih jedinica informacije, prikazana na nekom elektroničkom uređaju) dokumenta. Temeljna zadaća HTML jezika je uputiti web preglednik kako prikazati hipertekst dokument. Pri tome se nastoji da taj dokument izgleda jednako bez obzira o kojem web pregledniku, računalu i operacijskom sustavu je riječ. HTML nije programski jezik. Njime nije moguće izvršiti nikakvu operaciju, čak ni najjednostavniju operaciju zbrajanja ili oduzimanja. On služi samo za strukturni opis hipertekstualnih dokumenata.

HTML datoteke su obične tekstualne datoteke sa ekstenzijom *.html* ili *.htm*. HTML radi na „tag“ sistemu, odnosno, tagovi ili oznake su osnovne građevne jedinice HTML dokumenta. One opisuju kako će web stranica izgledati na web pregledniku. [7] Oznake su ono što odvaja sadržaj od HTML kôda. To su elementi HTML-a unutar kutnih zagrada, npr. *<head></head>*. Svaka HTML oznaka počinje znakom <, a završava znakom >. Zatvarajuća HTML oznaka kreira se na isti način kao i otvarajuća oznaka, ali se prije imena oznake dodaje i znak /. Ovi elementi nam omogućuju stvaranje osnovnog izgleda stranice, te govore web pregledniku što se treba prikazivati. [8] Osnovne oznake HTML dokumenta definirane su još od njegove prve verzije. Svaki HTML dokument preporučljivo je započeti *<!DOCTYPE>* oznakom, kojim se označava DTD (*Eng. Document-Type-Declaration)*, čime se definira točna inačica standarda HTML-a koji se koristi za izradu dokumenta. Nakon ovog elementa, *<html>* oznakom se označava početak HTML dokumenta. Unutar *<html>* elementa nalaze se

<*head*> i <*body*> element. Prvi navedeni predstavlja zaglavlje HTML dokumenta u kojemu se najčešće specificiraju jezične oznake HTML dokumenta, kao i sam naslov stranice. Dok se u <*body*> elementu kreira sadržaj HTML dokumenta.



Slika 5 - Osnovna struktura HTML dokumenta

Svakom novijom inačicom HTML standarda, dolazile su i nove oznake koje su povećavale njegove mogućnosti. Prvi javno dostupan opis HTML-a je dokument zvan „HTML tags“, prvi puta se spominje na internetu od strane Tim Bernes-a Lee-a 1991. godine, te opisuje 20 elemenata početnog dizajna HTML-a. Prva javno dostupna verzija HTML-a objavljena je 1993. godine. U to vrijeme, HTML je bio poprilično ograničen, te nije bilo čak ni mogućnosti za dodavanje slika u HTML dokument. [7]

Kroz godine i kroz revizije standarda, napredovao je i HTML. Trenutno aktualna je peta verzija HTML-a, nazvana HTML5 (objavljena 22.01.2008. godine, te dodatno ažurirana u listopadu 2014. godine). Cilj verzije je unaprijediti jezik uz podršku najnovijim multimedijskim značajkama. Dodatno, želi se osigurati laka čitljivost jezika od strane ljudi, a i računala, tj. Web preglednika. Uključene su mnoge sintaktičke značajke. Prva skupina odnosi se na obradu multimedijskih i grafičkih sadržaja. Sadrži oznake <*video*>, <*audio*>, <*canvas*>, od kojih je za ovaj rad najbitniji <*canvas*> element. Ovaj element se koristi za crtanje grafike na web stranici (uglavnom uz pomoć JavaScripta), i jedan je od najbitnijih za izradu igara na webu. Osim grafičkih elemenata, HTML5 donosi nove strukturne elemente za obogaćenje semantičkog sadržaja dokumenta (<*main*>, <*section*>, <*article*>, <*header*>, <*footer*>, <*aside*>, <*nav*>, <*figure*>). Dodani su novi atributi elemenata, neki atributi (i/ili elementi) su uklonjeni, a drugi (kao <*a/*>, <*cite*> i <*menu*>) su promijenjeni, redefinirani ili standardizirani. [9]

S povećanjem zahtjeva i razvojem tehnologije web-a, HTML-u se priključuju i „pomoćne“ tehnologije, koje ga dodatno upotpunjuju. Osnovne su CSS (koji detaljnije definira izgled stranice) i JavaScript (koji donosi novu razinu interaktivnosti na stranici). Obje tehnologije se pomoću odgovarajućih HTML oznaka unutar zaglavlja stranice dodaju na stranicu (bilo kao direktno uvedeni segmenti kôda odgovarajućeg jezika ili kao reference na vanjsku CSS/JavaScript datoteku).

4.2. CSS

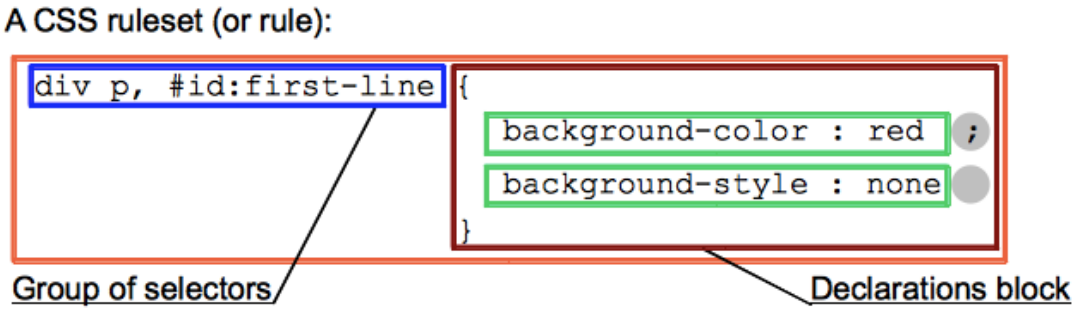
CSS (*eng. Cascading Style Sheets*) je stilski jezik koji se koristi za opisivanje prezentacije dokumenta pisanog jezikom za označavanje (HTML). On je temeljna tehnologija web-a, zajedno sa HTML-om i JavaScript-om.

CSS je dizajniran kako bi omogućio odvajanje prezentacije i sadržaja (izgled, boje, fontovi). Ovo razdvajanje poboljšava dostupnost sadržaja, pruža veću fleksibilnost i kontrolu u specifikaciji značajki stranice, omogućuje da više web stranica dijele oblikovanje određivanjem relevantnog CSS-a u zasebnoj .css datoteci, te smanjuje složenost i ponavljanje u strukturnom sadržaju. Riječ „cascading“, tj. Kaskadno dolazi iz određene prioritetne sheme kako bi se odredilo koje se stilsko pravilo primjenjuje ako više od jednog pravila odgovara određenom elementu.

CSS ima jednostavnu sintaksu i koristi brojne engleske ključne riječi kako bi odredio imena različitih stilskih svojstava. Dokument CSS-a se još naziva i stilski list, te sadrži popis pravila koje utječu na prikaz elemenata stranice. [10]

Svako pravilo ili skup pravila sadrže:

- jedan ili više selektora – odabiru elemente za koje se želi primijeniti ažurirana vrijednost entiteta (npr. Primjena CSS pravila na sve stavke HTML dokumenta)
- blok za deklaraciju – skup svojstava koji imaju postavljene vrijednosti kako bi ažurirali prikaz HTML sadržaja (npr. Širina elementa 50% širine roditeljskog elementa, pozadina postavljena na crveno)



Slika 6 - Struktura CSS pravila

Slika 6 - Struktura CSS pravila prikazuje CSS pravilo. Zelenim obrubom je prikazane CSS deklaracije. One služe za specificiranje promjene odgovarajućih svojstava HTML elementa. Svaka deklaracija se sastoji od dva dijela: Svojstva i vrijednosti nakon znaka :.

Više deklaracija tvori grupe deklaracija, tj. deklarativne blokove, koji su zatvoreni vitičastim zagradama. Svaka deklaracija unutar deklarativnog bloka (osim zadnje) mora biti odvojena znakom ;.

Prefiks deklaracijskom bloku (elementi ispred prve „lijeve“ vitičaste zagrade, {) primjenjuje promjene definirane deklaracijskim blokom na element/elemente specificirane ovim

prefiksom. Drugi naziv za navedeni prefiks je Selektor. Selektor je uzorak koji se poklapa sa određenim nazivom elementa stranice. Nad tim elementima stranice će se primijeniti promjene definirane unutar deklaracijskog bloka. Sveukupni naziv za selektore, deklaracijske blokove i deklaracije je CSS pravilo (na Slici označeno crvenom bojom). [11]

5. JavaScript

JavaScript, često skraćeno kao JS, je programski jezik visoke razine, također karakteriziran kao dinamičan, prototipni i jezik s više programskih paradigmi (npr. Javascript kod moguće je pisati proceduralnim načinom ili objektno orijentiranim). Uz HTML i CSS, JavaScript je jedna od tri osnovne tehnologije World Wide Weba. On omogućuje interaktivnost na web stranica i bitan je dio web aplikacija. Velika većina web stranica danas koristi JavaScript i svi glavni web preglednici imaju vlastiti JavaScript „motor“ za njegovo izvođenje.

Kao jezik s više paradigmi, JavaScript podržava stilove programiranja temeljene na događajima i funkcijama. Ima API (*Eng. Application-Program-Interface)* za rad s tekstom, nizovima, datumima, regularnim izrazima i osnovnom manipulacijom DOM-a (*Eng. Document-Object-Model)*. Početno biva implementiran samo na stranicama klijentovog web preglednika, danas su JavaScript pokretači implementirani u mnoge druge vrste host softvera (softveri čije se izvođenje vrši na serverskoj strani web servera, bazama podataka, čak i u programima za obradu teksta, npr. PDF).

Iako postoje jake vanjske sličnosti između JavaScripta i Jave, uključujući naziv jezika, sintaksu i odgovarajuće standardne biblioteke, jezici su drastično različiti. Na razvoj JavaScript-a su utjecali programski jezici kao što su Self i Scheme.

Od Svibnja 2017. godine, 94,5% od 10 milijuna najpopularnijih web stranica, koristilo je JavaScript. Njegova najčešća upotreba je dodavanje ponašanja HTML stranice na strani klijenta, također poznat kao Dynamic HTML (DHTML). Skripte su ugrađene ili uključene iz HTML stranica i vrše interakciju sa DOM-om stranice.

Neki primjeri upotrebe JavaScripta:

- Učitavanje nove stranice ili slanje podataka poslužitelju putem Ajax-a bez ponovnog učitavanja stranice (npr. Društvena mreža može dopustiti korisniku da objavi status bez napuštanja trenutne stranice)
- Animacija elemenata stranice, smanjivanje/povećanje vidljivosti, smanjivanje/povećanje veličine, rotacija, itd.
- Interaktivni sadržaj, npr. Igre i reprodukcija zvuka i videozapisa
- Validacija ulaznih vrijednosti web obrasca kako bi bili sigurni da su prihvatljivi prije slanja na poslužitelj

- Prijenos informacija o korisničkim čitateljskim navikama i aktivnosti pregledavanja na različitim web stranicama (ovo se često radi za web analizu, praćenje oglasa, personalizaciju weba itd.)

Budući da JavaScript kod može raditi lokalno na korisničkom pregledniku, preglednik može brže reagirati na korisničke radnje, tako da je aplikacija više responzivna. Nadalje, JavaScript kod može otkriti korisničke radnje koji sami HTML ne može, kao što su zapisi tipkanja. Aplikacije kao što je Gmail koriste tu opciju: mnogo je logike korisničkog sučelja napisano u JavaScriptu, a JavaScript šalje zahtjeve za informacijama poslužitelju.

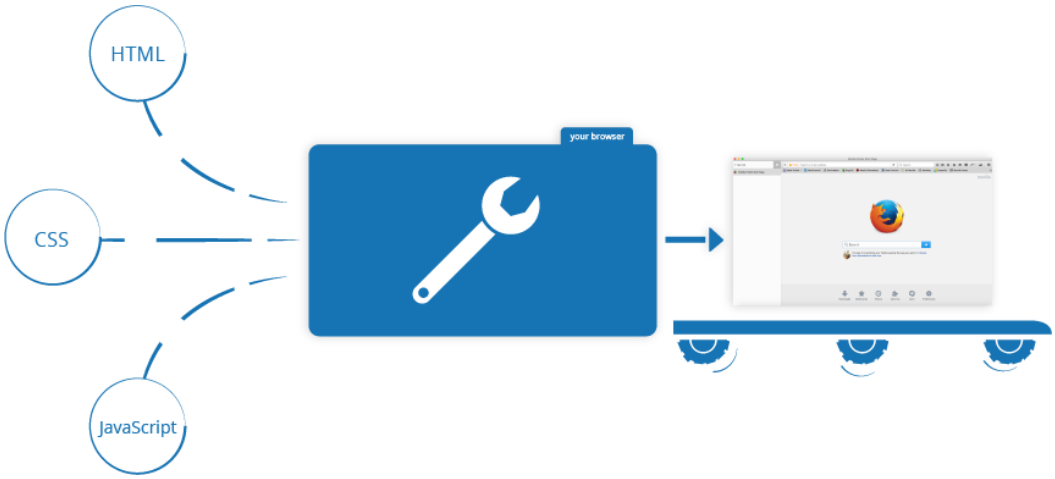
JavaScript motor (također poznat kao JavaScript interpreter ili JavaScript implementacija) je tumač koji tumači izvorni kod i izvršava skriptu u skladu s njim. Prvi JavaScript motor stvorio je Brendan Eich u Netscapeu, za Netscape Navigator web preglednik. Motor, pod nazivom Spider Monkey, implementiran je u C-u, a od tada je ažuriran (u JavaScript 1.5 verziju) u skladu sa ECMAScript 3 (Specifikacija skriptnih jezika, standardizirana od tvrtke ECMA u svrhu standardizacije JavaScript-a). Drugi je Rhino motor, primarno stvoren od strane Norris-a Boyd-a, je JavaScript implementacija u Javi. Rhino, poput SpiderMonkey-a, usklađen je sa ECMAScript 3.

Web preglednik je daleko najčešća okolina JavaScript domaćina. Web preglednici obično stvaraju „objekte domaćina“ koji predstavljaju DOM u JavaScriptu. JavaScript program ispituje i manipulira s ciljem dinamičke generacije web stranica. Budući da je JavaScript jezik za kojeg najpopularniji preglednici dijele podršku, postao je ciljni jezik za mnoge okvire drugih jezika. [12]

5.1 Javascript kod unutar web preglednika

Kada se web stranica učita u web pregledniku, pokrene se kod (HTML + CSS + Javascript) unutar izvršne okoline (trenutna kartica web preglednika). Web preglednik preuzima sve komponente stranice i kao rezultat daje web stranicu/aplikaciju. Kao što je u prethodnom poglavlju navedeno, Javascript motor je komponenta koja izvršava Javascript kôd na stranici. Ovo se događa nakon što su HTML i CSS komponente stranice učitane na stranicu. Razlog ovakvog načina rada je postavljanje strukture stranice prije nego što se Javascript funkcionalnosti krenu izvršavati, te time stranica radi brže.

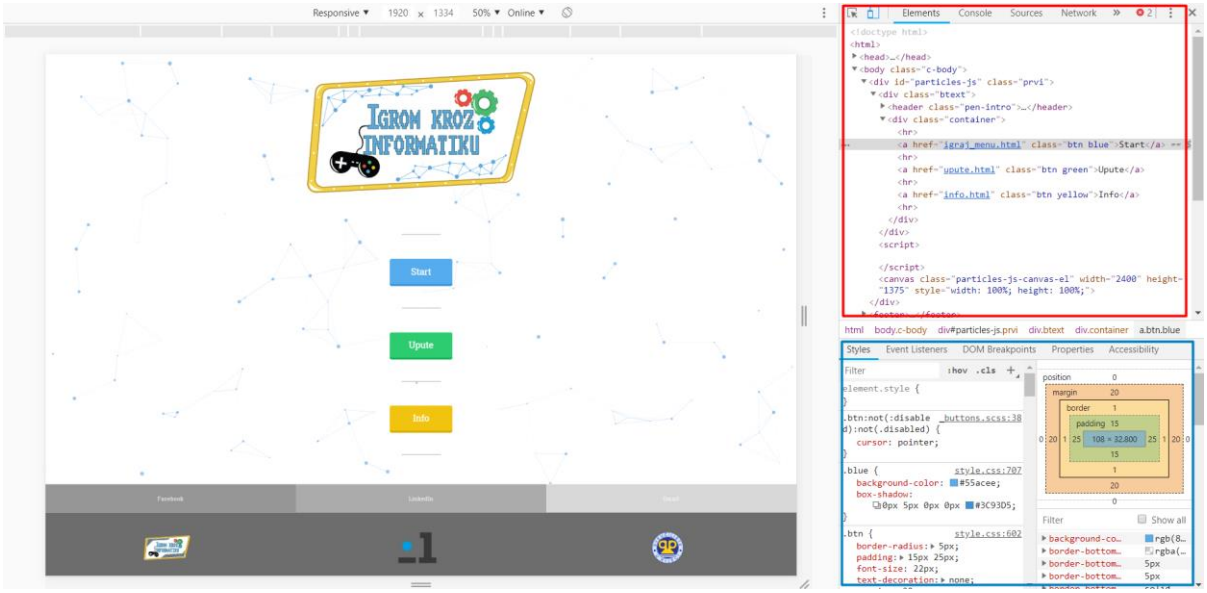
Dodatni, razlog je što je najčešća upotreba Javascript kôda dinamička izmjena HTML i CSS elemenata stranice. Ovime se stranica ažurira na korisničke unose, interakcije itd. Te je time iskustvo na stranici zabavnije, interaktivnije i više personalizirano korisniku. Svaka kartica preglednika ima svoje zasebno izvršno okruženje kôda. Kod na svakoj kartici radi potpuno odvojeno, te kod jedne kartice ne može izravno utjecati na drugu. [13]



Slika 7 - Izvođenje stranice

Javascript kod se može pisati u najobičnijem text editoru (npr. Windows Notepad). Pisanje kôda bazičnim tekst editorom je u redu za učenje osnovne sintakse jezika, no za pisanje kôda u profesionalnim svrhama, koriste se napredniji Javascript urednici teksta. Oni imaju značajke, funkcionalnosti i biblioteke posebno dizajnirane za ubrzanje i olakšanje pisanja izvornog kôda. Npr, olakšanje označavanja sintakse, strukturiranje kôda, korištenje gotovih funkcija itd. Urednik izvornog kôda može provjeriti sintaksu kôda odmah prilikom njegovog pisanja i upozoriti programera na određene greške. Greška kôda pisana u Notepadu nije vidljiva sve dok se stranica ne učita u web pregledniku. Postoje mnogi Javascript urednici teksta (Sublime tekst, Visual Studio Code, Atom, Brackets, Notepad++ itd.). Za potrebe ovog rada koristio se alat Atom. Poveznica na službenu stranicu editora je: https://atom.io/

Osim Javascipt editor, alati koji dodatno pomažu kod izrade web aplikacije su implementirani unutar samog web preglednika (tzv. developer alati). Svaki moderni web preglednik sadrži snažan paket alata za razvojne programere. Da bi se došlo do ovih alata na stranici, dovoljno je kliknuti desni klik miša i odabrati opciju inspect ili kraće, otvoriti ih uz pomoć tipke F12. Ovi alati rade niz stvari, od pregleda trenutno učitanog HTML-a, CSS-a i Javascript kôda kako bi se prikazalo koji dijelovi stranice su se učitali i koliko vremena im je trebalo da se učitaju. [14]



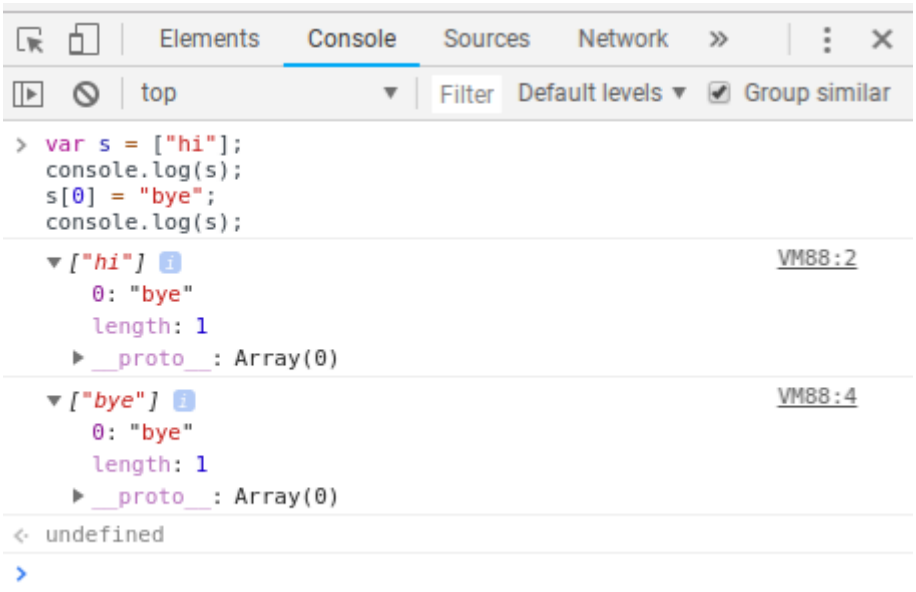
Slika 8 - Developer alati unutar web preglednika (Google Chrome)

Otvaranjem developer alata na web sjedištu ovog projekta vidljive su dvije njegove komponente (*Slika 8 - Developer alati unutar web preglednika (Google Chrome)*). Crveno označeno je tab elemenata u kojem je vidljiv HTML kod stranice. U ovom slučaju, selektiran je plavi startni gumb, zato se unutar taba elemenata označava njegova HTML oznaka unutar HTML dokumenta. Iz oznake je vidljivo da se klikom na njega otvara „*igraj_menu.html*“ stranica. Dodatno mu je pridodana klasa „*btn blue*“. U plavom označenom dijelu alata smješten je ekran CSS stilova selektiranog elementa, odnosno prikazuje stilove primijenjene na plavi start gumb. Npr. scrollanjem dolje možemo vidjeti stilove koji mu se primjenjuju preko prethodno spomenute „*btn blue*“ klase:

```
.blue {                    style.css:707
    background-color: #55acee;
    box-shadow:
        0px 5px 0px 0px #3C93D5;
}
.btn {                     style.css:602
    border-radius: 5px;
    padding: 15px 25px;
    font-size: 22px;
    text-decoration: none;
    margin: 20px;
    color: #fff;
    position: relative;
    display: inline-block;
    width: 160px;
}
```

Slika 9 - Stil plavog gumba

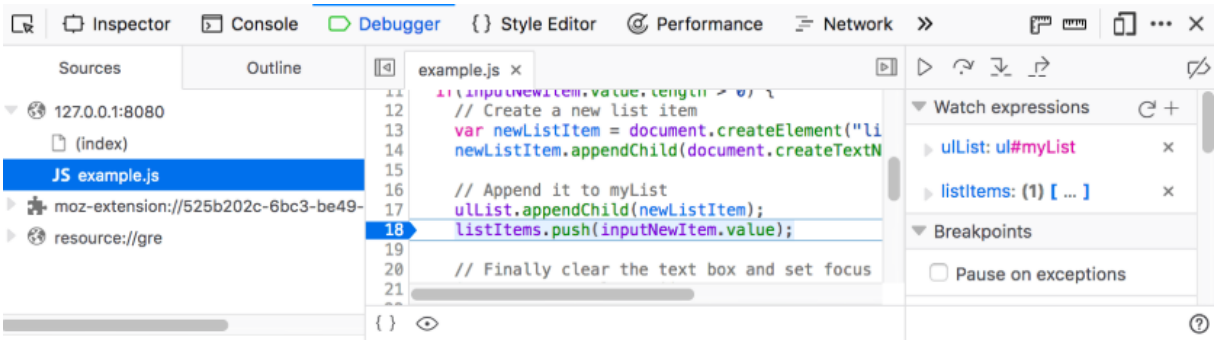
Uz pomoć navedenih sekcija developer alata, moguće je unutar Internet preglednika utjecati na HTML i CSS stilove stranice, te time vidjeti rezultat uživo na stranici (stranica je naravno, prikazana lijevo od alata). Osim ove dvije sekcije, iduće bitne su Javascript konzola i debugger. Konzola je vrlo koristan alat za ispravljanje grešaka unutar Javascript kôda koji ne funkcionira kako se očekuje. Ona omogućuje pokretanje linija JavaScript kôda direktno unutar stranice na kojoj je konzola i pokrenuta, i dodatno izvješćuje programera o pogreškama koje se dešavaju.



Slika 10 - Javascript konzola

Npr. na *Slika 10 - Javascript konzola* vidljiva je Javascript konzola. U konzoli je prvo deklarirana varijabla s koja je polje s jednim string elementom „hi“. Varijabla se potom ispisuje u konzoli uz pomoć console.log naredbe. Console.log naredba jedna je od najbitnijih naredbi prilikom ispravljanja grešaka unutar Javascript kôda. Općenito, ona ispisuje proslijeđenu vrijednost unutar svojih zagrada na Javascript konzolu. Ta vrijednost može biti varijabla „s“ kao u slučaju na *Slika 10 - Javascript konzola*, ali može biti i bilo šta drugo. Npr. polje, rezultat funkcije, tip varijable, objekt i njegove instance itd.

Ukoliko kod ima kakvu grešku, konzola i console.log naredba uvelike pomažu u njenom otkrivanju. Npr. Konzola bi mogla izbacivati grešku prilikom zbrajanja dva broja, s time da je jedan od tih brojeva deklariran kao string. Console log bi pomogao na način da se uz pomoć njega ispiše svaka od te dvije varijable. Slično kao i u primjeru sa *Slika 10 - Javascript konzola*, varijable bi se ispisale na sličan način, zajedno sa parametrima koji opisuju svaku. Za varijablu koja je broj, bilo bi vidljivo da ju ispisuje plavom bojom, dok bi varijablu deklariranu kao string ispisivalo crvenom bojom unutar navodnika. Na temelju ovoga, brže se dolazi do uzroka greške i greška se nakon toga u Javascriptu lako uklanja. Naravno, greške unutar kôda mogu biti i kompleksnije prirode (logička greška, tj. Program radi ali ne radi kako bi trebao). U tom slučaju pomaže Javascript debugger unutar web preglednika:



Slika 11 - Javascript debugger

Javascript debugger omogućuje praćenje vrijednosti varijabli prilikom samog izvođenja kôda. Moguće je i postavljanje prekidne točke u kodu gdje se izvođenje kôda zaustavlja i može se brže i lakše doći do prepoznavanja problema koji sprječavaju pravilno izvršavanje kôda. Lijevo na *Slika 11 - Javascript debugger* nalazi se Sources tab koji prikazuje listu datoteka vezane uz trenutno promatranu stranicu. Selektiranjem datoteke, otvara se i njen kod u centralnom prozoru debugger ekrana. Osim prikaza i pregleda kôda, u centralnom prozoru je moguće i postaviti prekidne točke izvođenja (koja je vidljiva na *Slika 10 - Javascript konzola*, te je označena na liniji 18). Kada izvođenje kôda dođe do ove definirane točke, ono se zaustavlja. U desnom dijelu debuggera nalazi se skup elemenata za praćenje promjena varijabli, lista dodanih prekidnih točaka i još dodatnih funkcija. Sve u kombinaciji su bitne za brzo i lako pronalaženje grešaka unutar Javascipt kôda.

6. Načini uključivanja Javascipt kôda u web projekt

Javascript se dodaje HTML-u slično kao što i CSS. CSS datoteke se pridodaju uz pomoć <link> elementa, dok se <style> oznaka koristi za dodavanje CSS stilova unutar HTML datoteke. Javascipt koristi <script> oznaku za dodavanje Javascipt kôda (bilo to internog unutar HTML datoteke ili kôda iz „vanjske“ Javascript datoteke).

6.1. Interno uključivanje

Kao što je navedeno, uključivanje Javascript kôda se vrši pomoću <script> oznake. Interno uključivanje podrazumijeva Javascript kod pisan unutar HTML datoteke. To je u redu kod učenja Javascripta ili internog uključivanja jako kratkog Javascript kôda.

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Homework</h1>

<p id="demo">JavaScript can change the style of an HTML element.</p>

<script>
function myFunction() {
    document.getElementById("demo").style.fontSize = "50px";
    document.getElementById("demo").style.color = "blue";
}
</script>

<button type="button" onclick="myFunction()">Click Me!</button>

</body>
</html>
```

Slika 12 - Interno uključivanje Javascript kôda

Na *Slika 12 - Interno uključivanje Javascript kodakôda* vidljiv je HTML dokument. On sadrži tijelo stranice označeno <body> elementom, unutar kojeg su tri elementa. Prvi je naslov unutar <h1> oznake sadržaja „Javascript Homework“, idući je paragraf teksta „Javascript can change the style o fan HTML element.“ Sa pridodanim ključem naziva „demo“. Zadnji element je gumb sa oznakom „Click me!“. Osim deklariranog „button“ tipa, sadrži i događaj na klik miša koji pokreće Javascript funkciju „myFunction()“. Interno uključen Javascript kod nalazi se unutar <script> oznaka te sadrži navedenu myFunction funkciju. Ona ne poprima nikakve ulazne varijable. Prilikom njenog pozivanja, pokreću se dvije selekcije elementa HTML stranice sa ključem „demo“, tj. Selektira se paragraf. A pomoću .style operatora se dodaju CSS stilovi za veličinu teksta na 50px i boju teksta na plavu. Da, CSS je moguće deklarirati i preko Javascripta na ovaj način, no inače se CSS piše u zasebnoj CSS datoteci.

6.2. Uključivanje eksterne Javascript datoteke

U praksi je ovo najčešći način uključivanja Javascripta u HTML. Razlog tome je što se time odvaja Javascript datoteka (također i CSS) od glavne HTML datoteke. Time je pregledavanje datoteka stranice lakše, što u konačnici garantira lakše održavanje. U realnom projektu sadržaji svih datoteka veliki i datoteka ima mnogo.

Odvajanje Javascript kôda sa primjera na *Slika 12 - Interno uključivanje Javascript kodakôda* bi započelo ponovno deklaracijom <script> elemena. No u ovom slučaju, on bi izgledao ovako:

<script src=“script.js“></script>

Unutar <script> oznaka trenutno nema ničega. No elementu je nadodan scr (*eng. Source*) oznaka s imenom „script.js“. Ovime je na trenutnu HTML stranicu uključena eksterna Javascript datoteka s navedenim nazivom. Kod iz prethodnog slučaja, nalazio bi se u ovoj datoteci. Time se postiže isti rezultat na samoj web stranici, no struktura datoteka je odvojena i više organizirana. Svi segmenti stranice (HTML, Javascript i CSS) nalaze se u svojim datotekama što olakšava čitanje i održavanje kôda, te pridodaje mogućnost ponovnog korištenja Javascript kôda na ostalim HTML stranicama web sjedišta. Npr. myFunction funkcija bi možda bila potrebna i na nekoj drugoj HTML stranici. Ako je Javascript uključen na stranicu interno, on vrijedi samo na toj stranici te stoga ne bi bilo moguće dohvatiti funkciju s nekom drugom HTML datotekom. Istu funkciju, bilo bi potrebno interno deklarirati. Ukoliko bi ova funkcija bila potrebna na još 20 stranica, isti postupak bi se trebao napraviti za svih 20 stranica. Ovakav način rada je mukotrpan, neorganiziran i loš za održavanje. Eksternim uvođenjem i deklaracijom myFunction funkcije, funkcija se deklarira samo jednom, te ju u svih 20 dokumenata uvodimo samo jednom linijom kôda napisanoj na prethodnoj stranici.

7. Javascript sintaksa

Sintaksa Javascripta je skup pravila koja definiraju pravilno strukturirani JavaScript program. Kao i svi ostali programski jezici, u Javascriptu je moguće korištenje svih poznatih aspekata programiranja. Od varijabli raznih tipova podataka, funkcija, operatora, uvjetovanih promjena, objekata itd. Osim navedenih, Javascript sadrži i tzv. Upravitelje događaja. Oni omogućuju da Javascript dohvaća i utječe na HTML elemente i/ili njihove CSS stilove. Uz pomoć njih se ostvaruje interaktivnost kod web stranica. Primjer Javascript događaja može biti kada korisnik klikne na određeni HTML element. Dohvaćen element i događaj na njemu pokreću određenu Javascript funkcionalnost koja utječe na prikaz stranice i elemente na njoj. Na *Slika 12 - Interno uključivanje Javascript kodakôda* Javascript događaj se nalazi unutar deklaracije gumba. Tj. to je onclick funkcija myFunction. Ona je događaj koji „sluša“ korisnikov unos, odnosno, klik na gumb. Ukoliko korisnik klikne na gumb, pokreće se definirana Javascript funkcija te se sadržaj stranice mijenja. Osim događaja na klik miša, postoje i drugi. Neki od češće korištenih su prikazani u sljedećoj tablici: [15]

Tablica 1 - Javascript događaji

Događaj	Opis događaja
onchange	HTML element je promijenjen
onclick	Korisnik je kliknuo na HTML element
onmouseover	Korisnik miče pokazivač miša preko HTML elementa
onmouseout	Korisnik je maknuo pokazivač miša dalje od granica HTML elementa

onkeydown	Korisnik je pritisnuo tipku na tipkovnici
onload	Preglednik je završio učitavanje web sjedišta

7.1. Stil pisanja

Što se tiče stila pisanja Javascript kôda, programski jezik je „case-sensitive“, odnosno, „osjetljiv“ na kapitalizaciju slova. Npr. Varijabla deklarirana kao ***var a = 5*** i druga varijabla deklarirana sa ***var A = 5*** nisu iste. Iste su sa stajališta vrijednosti jer su obje jednake 5 ali nisu iste sa stajališta pozivanja i korištenja njima. Ako se varijabla ***A*** uveća za 5, iznositi će 10. Nije isto pozvati varijablu ***a*** sa naredbom ***console.log(a)*** i očekivati da će rezultat biti 10. Rezultat ove naredbe ispisati će na konzoli web preglednika 5 a ne 10. Zaključno, može se reći da bi trebalo imena funkcija, varijabli ili bilo kojih drugih elemenata Javascripta pisati i koristiti pažljivo. Javascipt imena shvaća doslovno.

Javascript ignorira razmake, i prazne linije koje se pojavljuju unutar kôda. Programer može koristiti razmake, tabulatore, uvlačenje kôda, segmentiranje kôda na način koji on to želi. Time je kod u konačnici čitak i lak za razumijevanje.

Naredbe u Javascriptu obično završavaju s točka-zarez znakom kao što je to slučaj i u C-u, C++-u i Javi. Javacript omogućuje izostavljanje ovog znaka ako se svaka naredba nalazi u vlastitom retku. Javascript podržava komentare kôda na način identičan kao i C/C++. Stoga, svaki komad teksta između znaka // i kraja linije u kodu će biti tretiran kao komentar i bit će ignoriran kod izvođenja programa. Ovaj način komentiranja naziva se linijski komentar jer se proteže kroz samo jednu liniju unutar kôda. S druge strane, tekst između znakova **/*** i ***/** se smatra kao komentar, no u ovom slučaju moguće je komentirati kroz više linija kôda (tzv. višelinijski komentar). [16]

```
// This is a comment. It is similar to comments in C++

/*
* This is a multiline comment in JavaScript
* It is very similar to comments in C Programming
*/
```

Slika 13 - Javascript komentari

8. Bootstrap

Dodatna tehnologija koja ću se koristiti prilikom izrade stranice je Bootstrap. Bootstrap je besplatan front-end okvir (*Eng. Front-end framework* – skupina gotovog JavaScript kôda kojeg je moguće uključiti u vlastite projekte i programirati na brži način) otvorenog kôda za izradu web stranica i web aplikacija. Sadrži predloške temeljene na HTML-u i CSS-u za dizajniranje tipografije, obrasce, gumbe, navigaciju i ostale komponente sučelja. Zajedno s

time, sadrži i dodatna JavaScript proširenja. Za razliku od mnogih ranijih okvira, Bootstrap je fokusiran primarno razvojem front-end dijela aplikacije.

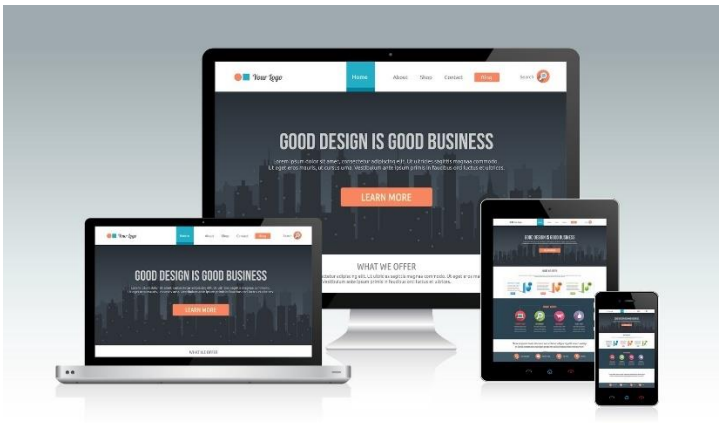
Glavni razlog korištenja Bootstrap-a za ostvarivanje responzivnosti stranice su njegove klase vezane uz raspored stranice. Ovaj raspored se naziva Grid layout (*raspored sadržaja korištenjem fluidnih mreža)*. Glavna svrha grid layout-a je responzivnost web sadržaja za sve uređaje kojim se isti pregledava. [17]

8.1. Responzivni web dizajn

Korištenje mobilnih uređaja za surfanje na webu raste rapidnom brzinom, nažalost, velika većina weba nije optimizirana za te uređaje. Mobilni uređaju često su ograničeni veličinom zaslona i zahtijevaju drugačiji pristup načinu postavljanja i rasporeda sadržaja na zaslonu. [18]

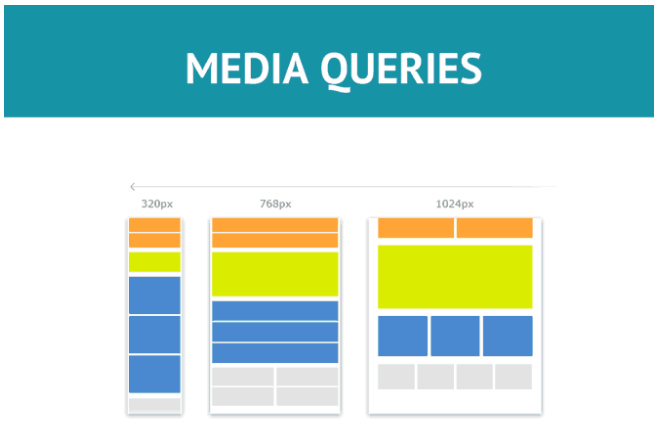
Osim pametnih telefona, postoji još mnogo različitih veličina zaslona uređaja. Od tableta, stolnih računala, igraćih konzola, televizora, pa čak i pametnih satova. Veličina zaslona uvijek se mijenja, stoga je važno da se web stranice mogu prilagođavati bilo kojoj veličini zaslona.

Prva ključna ideja koja stoji iza responzivnog dizajna jest korištenje takozvane fluidne mreže (*eng. Fluid grid*). Ne tako davno, korištenje fluidne mreže nije bilo tako popularno kao stvaranje stranica fiksne širine (Stranice koje su definirane fiksnim veličinama piksela, te zatim centrirane). Fluidne mreže idu nekoliko koraka izvan tradicionalnog izgleda stranice. Umjesto izgrade rasporeda stranice koja se temelji na fiksnim pikselima ili proizvoljnim postotnim vrijednostima, fluidna mreža je pažljivo osmišljena u smislu proporcija. Na ovaj način, kada je stranica „stisnuta“ na maleni zaslon mobilnog uređaja ili se proteže preko ogromnog zaslona, svi elementi mijenjaju širinu u odnosu jedni na druge. Problem dolazi kod malih ekrana npr., složeni raspored stranice u tri stupca neće dobro funkcionirati na malom mobilnom uređaju. Srećom, responzivni dizajn je riješio i ovaj problem korištenjem medijskih upita (*Eng. – Media queries)*.



Slika 14 - Responzivnost web stranice na različitim uređajima

Kod responzivnog dizajna, bitan podatak je širina zaslona koji se koristi za gledanje sadržaja. Medijski upiti koriste ovaj podatak od korisnika, te primjenjuju određene CSS stilove za svakog korisnika individualno. Npr. Ako je ekran veći od 1200px širine, primjenjuju se određeni stilovi i stranica izgleda na određeni način. Ako korisnik pak smanji prozor web preglednika (ili istu stranicu otvori svojim mobilnim uređajem), primjenjuju se CSS pravila koja promjenjuju izgled sjedišta da bi on bio pregledan i na manjim uređajima.



Slika 15 - Medijski upiti i njihovo djelovanje na izgled stranice

Fluidne slike su bitan dio sadržaja stranice, te se one također kao i tekst moraju mijenjati i prilagođavati dimenziji ekrana. Smanjenjem dimenzija ekrana, slike se moraju smanjivati bez da se deformiraju. Na njih se primjenjuju dodatni CSS stilovi da bi se ovaj uvjet i ostvario. [19]

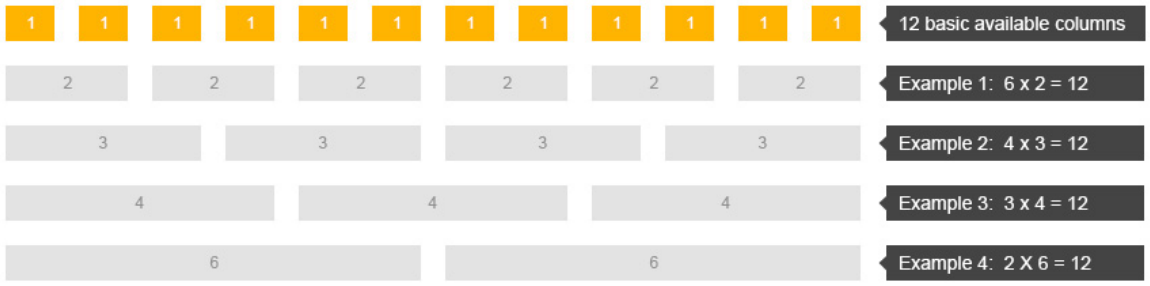
Zaključno može se reći da web stranica dizajnirana na responzivan način prilagođava svoj izgled pomoću fluidnih mreža, medijskih upita i fluidnih slika, odnosno mora zadovoljavati tri točke:

- Koncept fluidnih mreža poziva na mijenjanje veličine elemenata u relativnim jedinicama kao npr. Postotak, za razliku od fiksnih pikselnih vrijednosti
- Medijski upiti omogućuju da stranica koristi različita CSS pravila bazirana na karakteristikama uređaja koji pregledavaju tu stranicu (najbitnija informacija je širina zaslona)
- Fluidne slike su također definirane u relativnim jedinicama, te se time sprječava njihov prikaz van elementa u kojem se nalaze

8.2. Bootstrap grid layout

Bootstrap fluidna mreža je responzivna, mobile-first (razvoj stranice/aplikacije prvo za mobilne uređaje, tek na kraju za uređaje velikih dimenzija ekrana) fluidna mreža koja mijenja sadržaj do 12 stupaca s obzirom na način promjene veličina prozora, odnosno dimenzije ekrana kojim se pregledava sadržaj. Ona uključuje predefinirane klase za jednostavnu promjenu rasporeda i izgleda stranice. [20]

Ovih 12 standardnih stupaca pokriva punu širinu ekrana (neovisno o dimenziji ekrana). Moguće je sadržaj smjestiti u svaki od 12 ekrana, tako da npr. Stranica sadrži 12 paragrafa raspoređenih jedan pored drugog, po punoj širini ekrana. Svaki od ovih paragrafa ima mrežnu dimenziju 1, a njihov zbroj je 12; tj. Jedan redak sadržaja Bootstrap stranice zbrojno uvijek daje broj 12. Ukoliko neki element prelazi sa svojom dimenzijom ovaj broj, element se pozicionira u novi redak stranice. Jedinične stupce je moguće spajati i time dobivati veće ćelije. Na ovaj način definiramo raspored elemenata na stranici.



Slika 16 - Bootstrap mreža i prikaz njezinih 12 stupaca u 5 redaka

Bootstrap fluidna mreža sadrži 4 glavne klase:

- **xs** (za telefone – zaslone manje od 768px širine)
- **sm** (za tablete – ekrana jedna ili veća od širine od 768px)
- **md** (za manje laptope – ekrani jednaki ili veći od 992px širine)
- **lg** (za veća prijenosna računala i stolna računala – ekrani jednaki ili veći od 1200px širine) [21]

Uz pomoć ovih klasa za različite ekrane, moguće je stvoriti stranicu koja ima definiran izgled za svaku od dimenzija ekrana.

Dodatno, bitna pravila korištenja Bootstrap mreže su sljedeća:

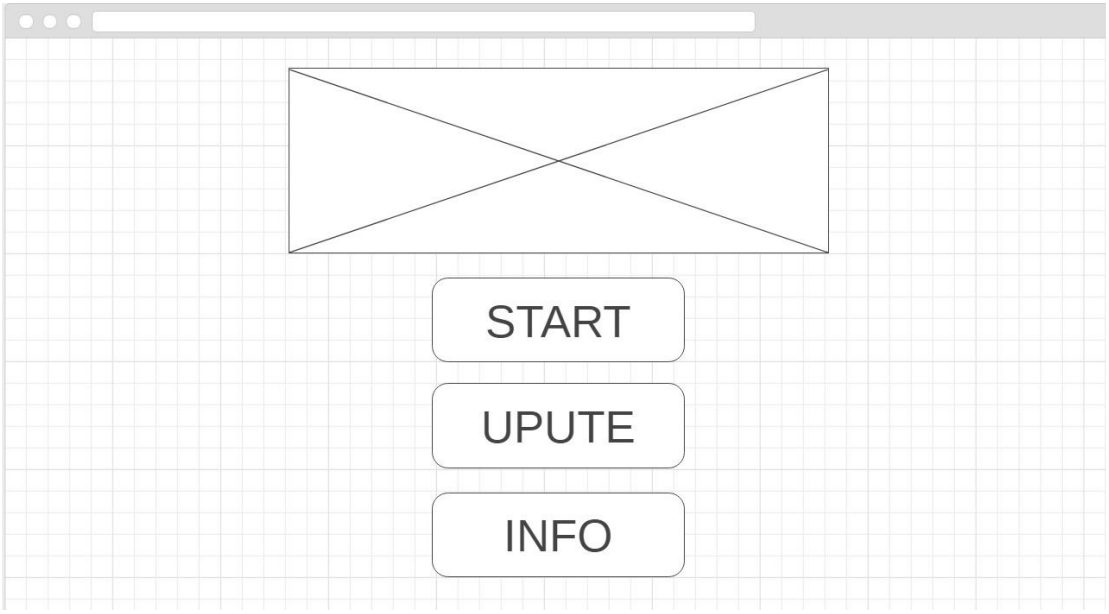
- Redci moraju biti postavljeni unutar kontejner klase (eng. Container class) fiksne širine ili fluidne širine za pravilno poravnanje i ispune
- Redci se koriste za izradu vodoravnih grupa stupaca
- Sadržaj stranice potrebno je smjestiti u stupce, a samo stupci mogu biti neposredna djeca redaka

- Predefinirane klase mreže poput .row i .col-xs-4 dostupne su za brzu i jednostavnu izradu rešetkastih rasporeda
- Stupci stvaraju žlijebove (Eng. Gutter – praznine između sadržaja stupaca, tj. Bootstrap preddefinirani razmaci između sadržaja). Razmak je u redovima za prvi i posljednji stupac pomaknut negativnom marginom
- Negativna margina je razlog zašto je sadržaj unutar mreže poravnat sa sadržajem van mreže
- Stupci mreže izrađeni su određivanjem broja do 12 raspoloživih stupaca. Npr., tri jednaka stupca mreže koristit će klasu .col-xs-4 (stupac koji na uređajima malih dimenzija pokriva širinu 4; s obzirom da ih je 3, njihov zbroj je 12 (4 + 4 + 4 = 12) i prostirat će se preko cijele širine ekrana malih uređaja)
- Ako je više od 12 stupaca smješteno unutar jednog retka, svaka grupa dodatnih stupaca će se kao jedna jedinica prebaciti u novi redak stranice
- Klase mreže primjenjuju se na uređaje s širinama zaslona koji su veći ili jednaki veličinama točke preopterećenja, a nadjačavaju klasu rešetki usmjerenih na manje uređaje. Stoga npr. Primjeno bilo koje .col-md-* klase elementu ne samo da će utjecati na njegov stil na srednjim uređajima već i na velikim uređajima ako ne postoji definirana klasa .col-lg-* [20]

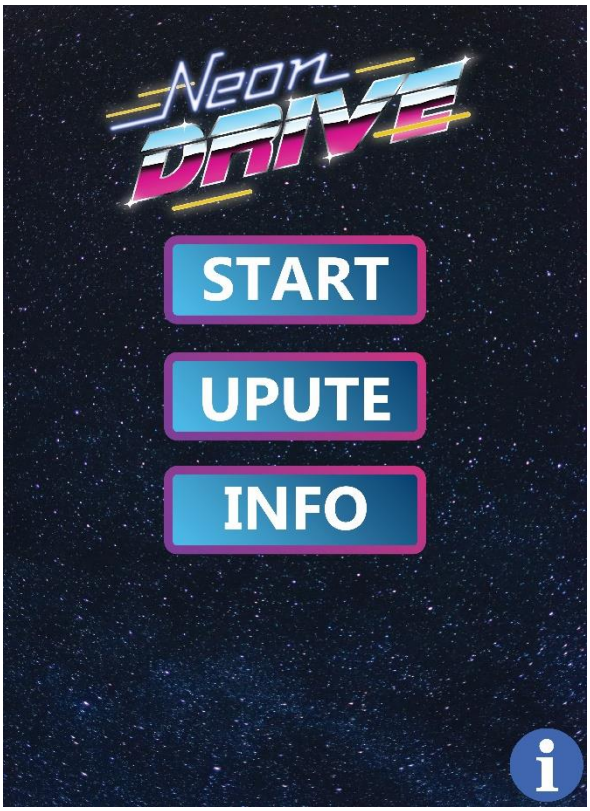
9. Opis izrađene interaktivne stranice s igrama

Od spomenutih tehnologija, HTML5 će se koristiti za strukturni dio stranice. Uz pomoć <canvas> oznake raditi ću HTML okvir za GBL igre. Same igre unutar okvira, zajedno sa ostalim interaktivnim elementima stranice, bit će izvedene sa JavaScriptom dok će prezentacijski dio stranice biti izveden sa CSS-om. Dodatno, za ostvarivanje responzivnosti stranice, koristi se Bootstrap za definiciju grid-ova, poravnanja i slično.

S obzirom da je stranica namijenjena za edukativne svrhe djeci od prvog do četvrtog razreda, izgled i navigacija stranice bit će vrlo jednostavni. Primjer skice početne stranice vidljiv je na Slici 17. Pri vrhu stranice biti će općeniti logo stranice, a ispod njega gumbi, redom nazvani „Start“, „Upute“ i „Info“. Klikom na gumb „Start“, prelazi se na zaslon gdje odabiremo igru (jer će biti izrađeno više edukativnih igara), te dalje klikom na željenu, prelazi se u izvođenje same igre. Gumb „Upute“ sadržavat će upute za svaku pojedinu igru stranice. Dodatno, odgovarajuću uputu igre, biti će moguće pročitati unutar prozora gdje se trenutno izvodi igra, a gumb „Info“ će korisnika odvesti na informacijsku stranicu ovog projekta. U donjem desnom kutu vidljiv je gumb informacija. Ovakav i slični gumbi, biti će korišteni „dublje“ u stranici. Dodatni mogu biti npr. Povratak na stranicu prije/početnu stranicu, tj. Navigacijski gumbi. Iduće dvije slike prikazuju ideju za početnu stranicu projekta. Prva je mockup (prototip koji prikazuje dio funkcionalnosti sustava i omogućuje testiranje dizajna, dizajneri koriste mockup modele za prikupljanje povratnih informacija od korisnika), dok je druga prikazuje primjer finalnog izgleda početne stranice (rađeno u Adobe Illustrator-u).



Slika 17 - Mockup početne stranice



Slika 18 - Skica početne stranice

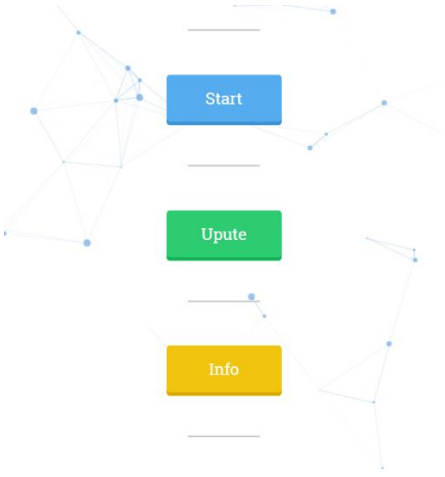
9.1. Početna stranica

Početna stranica rađena je prema predlošku opisanom u prethodnom poglavlju. Na vrhu stranice u centru se nalazi logo projekta. Logo je izrađen u Adobe Illustrator programu. Veći prikaz logotipa:



Slika 19 - Veliki logo početne stranice

Osim glavnog logotipa, također je kreirana i manja verzija istog koji se koristi u podnožju stranice (*eng. footer*) kao gumb za povratak na Home stranicu. Ispod logotipa se nalaze navigacijski gumbi za određene dijelove aplikacije. Gumbi su iste dimenzije, različitih boja, te svaki od njih ima efekt na klik i prijelaz mišem. Prijelazom miša iznad gumba, isti mijenja nijansu svoje odgovarajuće boje u svjetliju nijansu. Dok se klikom na gumb pokreće animacija klika, odnosno, gumb se "utisne" u ekran. Klikom na svaki gumb, prelazi se na istoimenu html stranicu sa odgovarajućim sadržajem.. Dodatno, pozadina ovog „glavnog" dijela stranice je animirana. Za nju je korištena particles.js skripta.

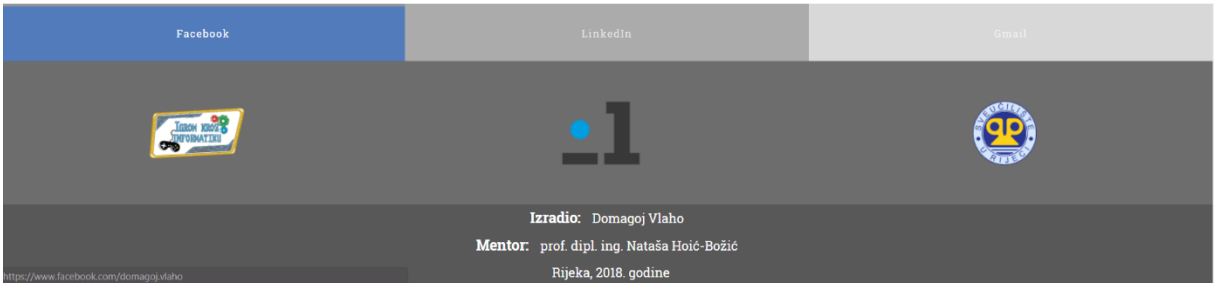


Slika 20 - Navigacijski gumbi početne stranice

Ispod gumba na početnoj stranici smješteno je podnožje stranice (*eng. footer*). Podnožje je sastavljeno od tri komponente (linkovi društvenih mreža, linkovi u obliku ikona i opće informacije o projektu). Linkovi društvenih mreža su pozicionirani horizontalno, "prilijepljeni" jedan na drugoga (rastegnuti po dužini ekrana sa paddingom i marginama na nuli). Svaki od njih je sive boje u različitim nijansama. Gumbi podnožja vode na društvene

mreže: Facebook, LinkedIn i Gmail. Prijelazom miša nad svakim od njih, mijenja se boja gumba u boju po kojoj je društvena mreža poznata (Facebook, LinkedIn plave i Gmail crvena). Klikom na svaki od njih, odlazi se na moje profile na svakoj od navedenih društvenih mreža.

Nakon prvog dijela podnožja (linkovi društvenih mreža) vidljiv je dio s linkovima u obliku ikona. Prva je prethodno spomenut manji logo projekta. Klikom na njega vraćamo se na početnu stranicu. Pored njega smješten je logo Odjela za Informatiku Sveučilišta u Rijeci. Klikom na njega, odlazi se na stranice OIRI, a na kraju je gumb Riječkog sveučilišta koji vodi na stranice sveučilišta u Rijeci. Cijelo podnožje ponavlja se i vidljivo je na svakoj stranici projekta. Konačno, na dnu stranice je jednostavno stilizirani HTML paragraf koji navodi opće informacije o projektu.



Slika 21 - Podnožje stranice

9.2. Informacijska stranica

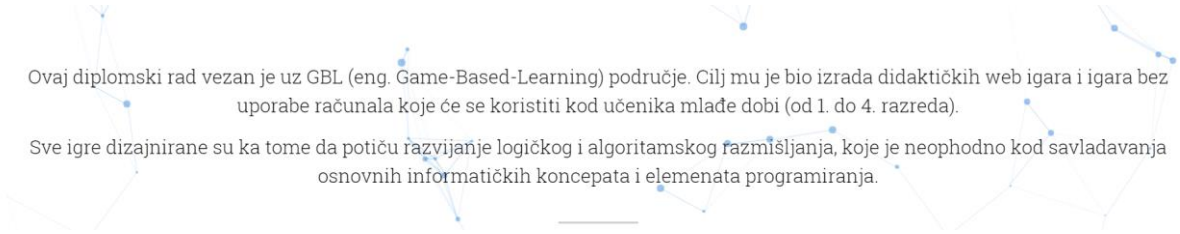
Informacijska stranica, kao što i sam naziv govori, sadrži informacije koje se odnose na cjelokupni projekt. Podijeljena je u 5 dijelova, od čega je prvo zaglavlje (*eng. Heading*):



Slika 22 - Zaglavlje stranice

Zaglavlje je vrlo jednostavno. To je *<header>* element html-a sa postavljenom sivom pozadinom i maksimalnom širinom s obzirom na dimenziju ekrana. Unutar *<header>* elementa dodana je manja verzija logotipa pozicioniranog na lijevi kraj *<header>* okvira, zajedno sa pomakom od lijevog ruba ekrana za 5% ukupne širine ekrana. Također, sa ovim stilom iz css-a, pridodana je i opcija da ako korisnik prijeđe pokazivačem preko slike, kursor se pretvara u ikonu kažiprsta.

Sadržaj između zaglavlja i podnožja je smješten unutar Bootstrap kontejnera. Idući dio info stranice sadrži odlomak općih informacija o projektu. On se nalazi unutar Bootstrap stupca definiranog da za sve ekrane pokriva veličinu od 12, odnosno, pri svim dimenzijama ekrana, ovaj će se sadržaj prikazivati preko cijele dužine ekrana.



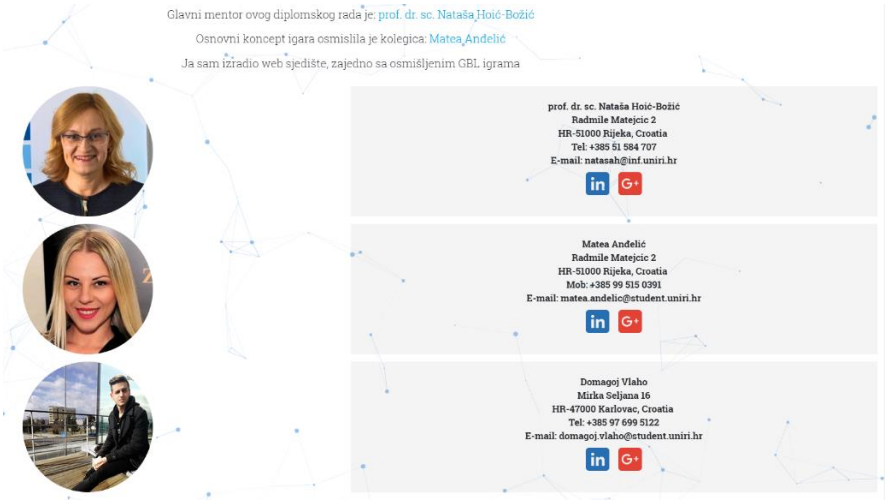
Slika 23 - Općeniti opis projekta

Idući dio stranice dobro prikazuje prednost kod jednostavnosti i organizacije sadržaja pomoću Bootstrap okvira. U ovom slučaju, imamo dva Bootstrap retka (lijevi za paragraf korištenih tehnologija i desni za slike). Elementi su podijeljeni i „slažu“ se s obzirom na veličinu ekrana koju korisnik koristi.



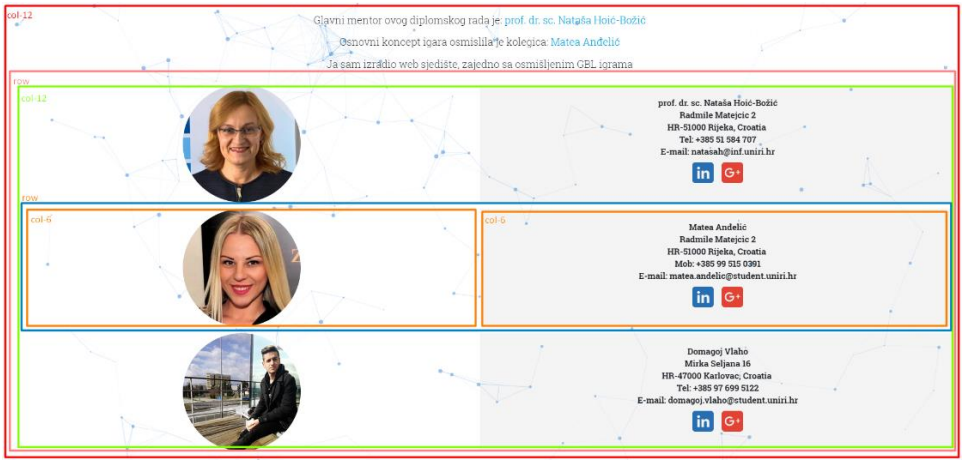
Slika 24 - Grid korištenih tehnologija i slika

Sljedeći dio informacijske stranice prikazuje sudionike koji su bili uključeni u stvaranju ovog projekta:



Slika 25 - Sudionici projekta

Iz *Slika 25 - Sudionici projekta* vidljivo je da je sekcija sudionika podijeljena na dva dijela. Prvi dio je paragraf sa tri natuknice u kojima se spominju svi sudionici, zajedno sa opisom njihovih uloga u projektu. Drugi je mreža koja prikazuje avatar slike svakog sudionika, te „kartice“ sudionika sa kontaktnim informacijama. Cijela sekcija sudionika (paragraf + avatari + kontaktne informacije) smještene su unutar Bootstrap retka veličine 12, tj. Svi elementi su razvučeni preko cijele dužine ekrana. Grafički bi mogli prikazati opisani grid ove sekcije na sljedeći način:

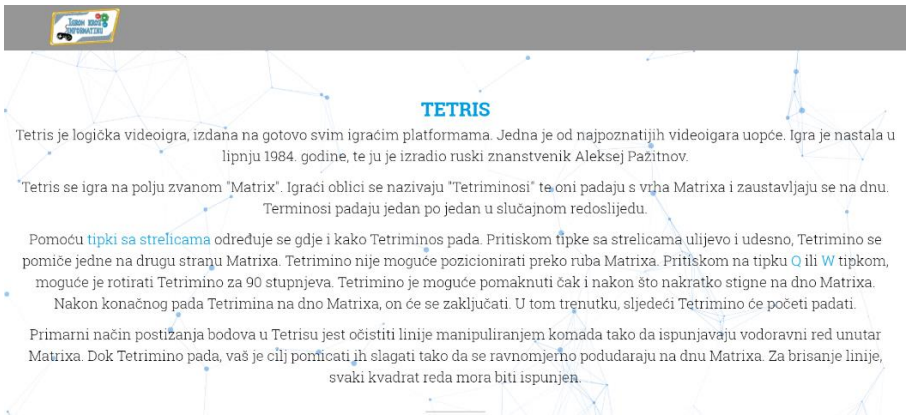


Slika 26 - Bootstrap grid sekcije sudionika

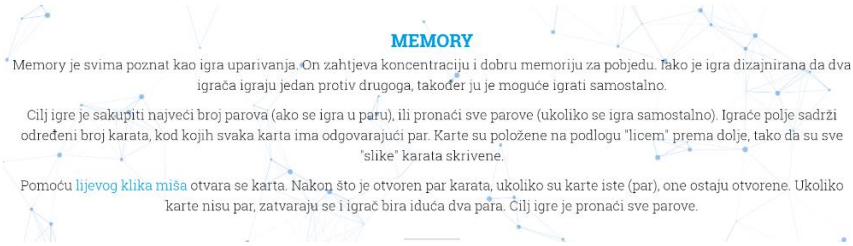
Ispod sekcije sudionika, nalazi se prethodno opisano podnožje stranice koje je isto na svakoj stranici projekta.

9.3. Stranica s uputama za igre

Klikom na gumb „Upute“ sa početnog zaslona, prelazi se na stranicu sa uputama za igre. Stranica je koncipirana identično kao i stranica sa informacijama, ali na jednostavniji način jer je njen sadržaj samo tekstualni opis pojedinih igara.



Slika 27 - Zaglavlje uputa + Tetris opis

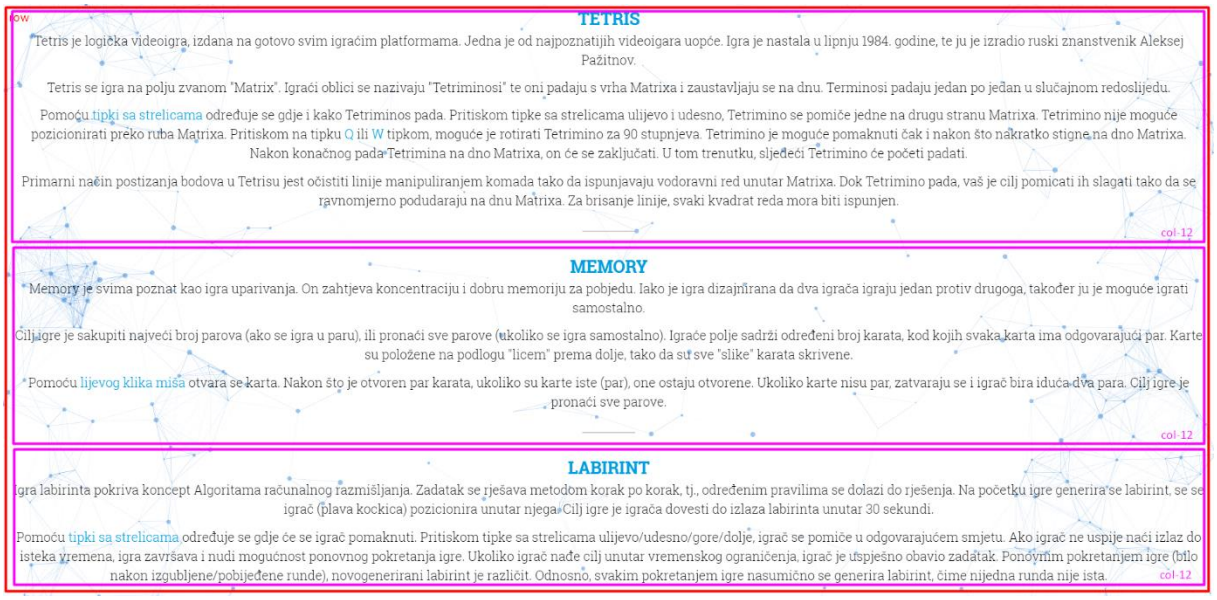


Slika 28 - Memory opis



Slika 29 - Opis Labirinta + podnožje

Zaglavlje i podnožje stranice je isto, pa ga nije potrebno opisivati. Bootstrap grid za glavni sadržaj stranice je definiran jednostavno. Bootstrap redak okružuje cijeli sadržaj. Redak je podijeljen u tri stupca, veličine 12. Grafički prikaz grid-a je sljedeći:



Slika 30 - Bootstrap grid uputa

Unutar paragrafa , neke su riječi obojane plavom bojom. One označavaju kontrole, tj. Tipke koje se koriste kod igranja odgovarajuće igre.

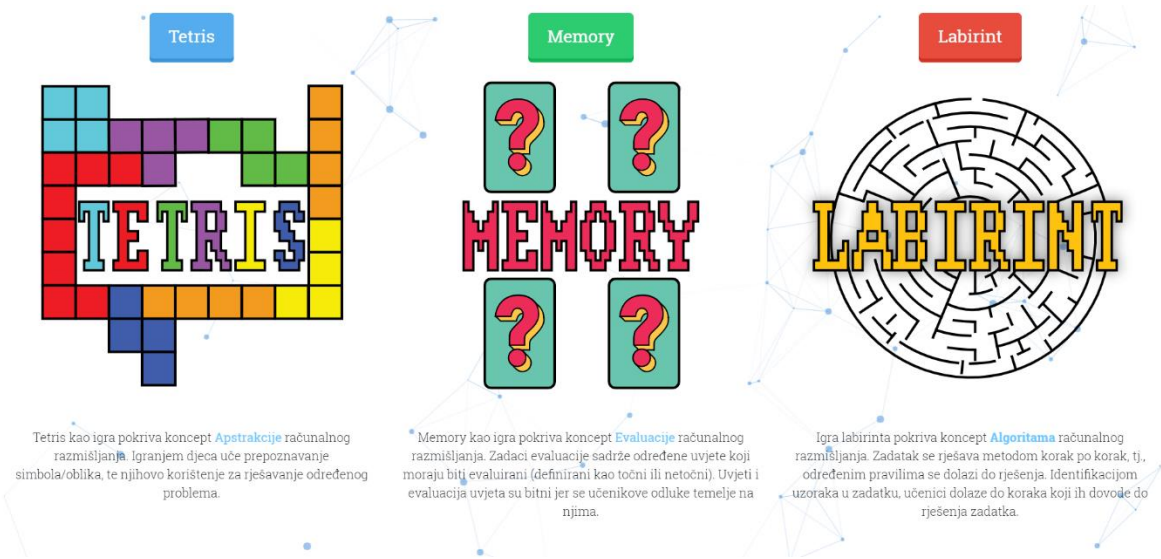
9.4. Stranica za selekciju igre (“meni” za odabir igre)

Zaglavlje stranice za selekciju igre koju korisnik želi igrati razlikuje se od zaglavlja prethodno opisanih jedino sa novododanim “X” gumbom. Gumb se nalazi na desnoj strani zaglavlja. Klikom na gumb vraćamo se na prethodnu stranicu. Selekcijom bilo koje od igara i ulaženjem “dublje” u stranicu, ovaj gumb će uvijek postojati u headeru i omogućavat će povratak za “jedan korak” unatrag na stranici.



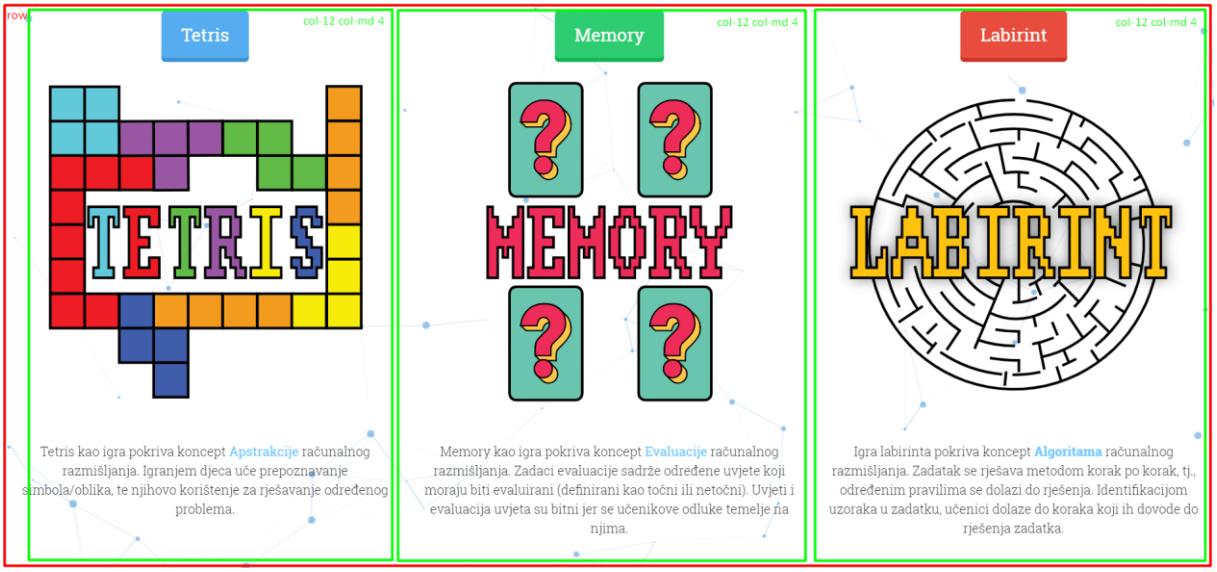
Slika 31 - Zaglavlje stranice selekcije igara

Glavni dio stranice za selekciju igre podijeljen je u tri dijela (po jedan za svaku od tri napravljene didaktičke igre).



Slika 32 - Glavni dio stranice za selekciju igara

Grid je sastavljen od jednog reda, u kojem se nalaze tri stupca (po jedan za svaku igru). Svaki stupac ima definiranu širinu od 4 jedinica (col-4).. U ovoj situaciji, sva tri retka mreže će se prostirati cijelom dužinom ekrana jer je zbroj njihovih vrijednosti 12, odnosno, puna širina prozora definirana Bootstrap klasama. Za manje ekrane, stupci će se rasporediti "jedan na drugog". Unutar jednog stupca nalazi se gumb sa nazivom igre (čijim klikom odabiremo tu određenu igru i prelazimo u dijelove stranice gdje se ta igra i odvija), logotip igre i kratki tekstualni opis što se uči igranjem određene igre. Plavo obojane riječi označavaju koncepte računalnog razmišljanja koje određena igra pokriva. Sva tri logotipa izrađena su u Adobe-ovom Illustrator programu. Zaglavlje stranice je identično kao i sva ostala zaglavlja web sjedišta.



Slika 33 - Bootstrap grid stranice za selekciju igre

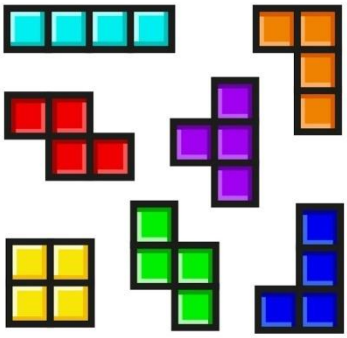
9.5. Tetris stranica

Prva igra koja je napravljena za potrebe rada je Tetris. Tetris je logička videoigra, izdana na gotovo svim igraćim platformama i jedna je od najpoznatijih videoigara uopće. Igra je nastala u lipnju 1984. godine, te ju je izradio ruski znanstvenik Aleksej Pažitnov. Tetris se igra na polju zvanom "Matrix". Igraći oblici se nazivaju "Tetriminosi" te oni padaju s vrha Matrixa i zaustavljaju se na dnu. Terminosi padaju jedan po jedan u slučajnom redoslijedu.



Slika 34 - Službeni Tetris logo

Tetris kao igra pokriva koncept apstrakcije računalnog razmišljanja. Igranjem djeca uče prepoznavanje simbola/oblika, te njihovo korištenje za rješavanje određenog problema. Igra je rađena prema video tutorialu (https://www.youtube.com/watch?v=H2aW5V46khA) za početničko učenje JavaScripta na koju sam naknadno dodao dva teža načina igre i stilski promijenio izgled da se uklapa u dizajn stranice.



Slika 35 - 7 elemenata (Tetriminosa) Tetris igre

Igra se pomoću tipki sa strelicama kojima se određuje gdje i kako Tetriminos pada. Pritiskom tipke sa strelicama ulijevo i udesno, Tetrimino se pomiče jedne na drugu stranu Matrixa. Tetrimino nije moguće pozicionirati preko ruba Matrixa. Pritiskom na tipku Q, W ili strelicu gore, moguće je rotirati Tetrimino za 90 stupnjeva. Tetrimino je moguće pomaknuti čak i nakon što nakratko stigne na dno Matrixa. Nakon konačnog pada Tetrimina na dno Matrixa, on će se zaključati. U tom trenutku, sljedeći Tetrimino će početi padati.

Primarni način postizanja bodova u Tetrisu jest očistiti linije manipuliranjem komada tako da ispunjavaju vodoravni red unutar Matrixa. Dok Tetrimino pada, cilj je pomicati ih slagati tako da se ravnomjerno podudaraju na dnu Matrixa. Za brisanje linije, svaki kvadrat reda mora biti ispunjen. Ukoliko je red ispunjen, on se „lomi“, nestaje, na njegovu poziciju padaju preostali Tetriminosi (ukoliko ih ima) i igrač dobiva 10 bodova. Nakon toga, igra se nastavlja i novi Tetrimino kreće padati. Tetrimina ukupno ima 7, te se svaki novi Tetrimino određuje slučajnom vrijednosti.

```
</header>
<body class="c-body">
  <div id="particles-js" class="igra1">
    <script type= "text/javascript" src= "particles.js"> </script>
      <script type= "text/javascript" src= "app.js"> </script>
    <div class="btext">
      <div id="igra1Container" class="container-fluid justify-content-center">
        <div class="row" d-flex align-items-center justify-content-center>
          <div class="col-12 " id="canvasPozadina" style="">
            <div class="row">
              <div class="col-12">
                <div id="score"></div>
                <select id="speedSelect" onchange="speed()">
                  <option value="beginner">Lagana</option>
                  <option value="intermediate">Classic</option>
                  <option value="hard">Teška</option>
                </select>
              </div>
            </div>
              <canvas id="tetris" width="240" height="400"></canvas>
              <script src="tetris.js"></script>
              </div>
          </div>
        </div>
      </div>
    </div>
```

Slika 36 - Tetris HTML

Unutar glavnog igraćeg prostora, vidljive su tri komponente. Prva je score div element koji prikazuje trenutni rezultat igrača. Za svaki slomljeni redak, igrač dobiva 10 bodova, kad terminosi dotaknu vrh igraćeg prostora, bodovi se resetiraju te igra započinje opet. Ispod brojača bodova nadodana je selekcija težine igre. Postoje tri: početnička, klasična i teška. Kod početničke težine, terminosi padaju sporo (jedna sekunda po koraku pada), srednja se ubrzava na 700 milisekundi, dok najteži nivo igre ubrzava na 400 milisekundi. Treći dio prozora je samo igraće polje Tetrisa smješteno u canvas element.

```
const canvas = document.getElementById("tetris");
const context = canvas.getContext("2d");

context.scale(20, 20);
```

Slika 37 - Dohvaćanje canvas elementa

Kod Tetrisa započinje dohvaćanjem canvas elementa i deklaracijom istog zajedno sa kontekst varijablom (*Slika 37 - Dohvaćanje canvas elementa)*. Osim toga, sa canvas metodom context.scale povećavamo omjer crtanja 20 puta.

```
const colors = [
  null,
  "#ff2580",
  "#25c8ff",
  "#25ff80",
  "#dc32e5",
  "#ff8e0d",
  "#ffe44b",
  "#326be5",
];

const arena = createMatrix(12, 20);

const player = {
  pos: {x:0, y:0},
  matrix: null,
  score: 0,
}
```

Slika 38 - Globalne varijable

Osim dohvaćanja canvas elementa, deklariraju se i globalne varijable (varijable kojima je moguće pristupiti iz bilo kojeg dijela kôda). Deklariraju se boje svih 7 Tetriminosa, matrica igraćeg polja i objekt igrača sa početnim pozicijama i resetiranim rezultatom.

```
function update(time = 0){
  const deltaTime = time - lastTime;
  lastTime = time;
  dropCounter += deltaTime;
  if (dropCounter > dropInterval){
    playerDrop();
  }
  draw();
  requestAnimationFrame(update);
}
```

Slika 39 - Update funkcija

Igra započinje pokretanjem update funkcije koja prima varijablu vremena postavljenim na nulu. Delta time služi za odbrojavanje, na temelju čega se uvećava dropCounter varijabla. Ukoliko je dropCounter varijabla veća od dropInterval vremena, pokreće se playerDrop funkcija kojom Tetrimino pada za jedan korak. Nakon toga se ponovno nacrta igraće polje s novom pozicijom i update funkcija se ponovno pozove. Ove vremenske varijable deklarirane su ovako:

```
let lastTime = 0;
let dropCounter = 0;
let dropInterval  = 1000;

function speed(){
  var x = document.getElementById("speedSelect").value;
  if(x === "beginner"){

      dropInterval = 1000;
      return dropInterval;
  }else if(x === "intermediate"){

      dropInterval = 700;
      return dropInterval;
  } else if(x === "hard") {

      dropInterval = 400;
      return dropInterval;
  }

}
```

Slika 40 - Vremenske varijable

LastTime i dropCounter varijable su prvotno deklarirane kao nula, dok je dropInterval postavljen na 1000 milisekundi. Tj., kada se igra prvi puta pokrene, ona će se krenuti izvoditi u laganom načinu igre gdje Tetrimino pada svakih 1000 milisekundi, odnosno, korak pada je jedna sekunda. Funkcija speed poziva se na odabir selekcijskog elementa težine igre. Za selektiranu težinu igre, dropInterval varijabla se postavi na odgovarajuću vrijednost i vraća se kao novo definirana dropInterval varijabla, čime se utječe na izvođenje igre.

```
function draw(){
  context.fillStyle = "#000";
  context.fillRect(0, 0, canvas.width, canvas.height);

  drawMatrix(arena, {x: 0, y: 0});
  drawMatrix(player.matrix, player.pos);
}

function drawMatrix(matrix, offset){
  matrix.forEach((row, y) =>{
    row.forEach((value, x) => {
      if (value !== 0){
        context.fillStyle = colors[value];
        context.fillRect(x + offset.x,
                 y + offset.y,
                 1, 1);
      }
    });
  });
}
```

Slika 41 - Funkcije crtanja

Draw funkcija ispunjava igraće polje crnom bojom i poziva drawMatrix funkciju dva puta (za element matrice igraćeg polja i igrača). Igrača matrica i igrač se definiraju i pozicioniraju unutar context elementa (tj. canvasa ili 2D polja) uz pomoć drawMatrix funkcije.

```
function createMatrix(w, h){
  const matrix = [];
  while (h--){
    matrix.push(new Array(w).fill(0));
  }
  return matrix;
}
```

Slika 42 – Igraća matrica

Igraća matrica se definira sa createMatrix funkcijom i ispunjava svaku ćeliju polja s nulom. Razlog ovoga je definicija Tetris elemenata:

```
function createPiece(type){
  if (type === "T"){
    return [
      [0, 0, 0],
      [1, 1, 1],
      [0, 1, 0],
    ];
  } else if(type === "O"){
    return [
      [2, 2],
      [2, 2],
    ];
  } else if(type === "L"){
    return [
      [0, 3, 0],
      [0, 3, 0],
      [0, 3, 3],
    ];
  } else if(type === "J"){
    return [
      [0, 4, 0],
      [0, 4, 0],
      [4, 4, 0],
    ];
```

Slika 43 - igraći elementi

Iz *Slika 43 - igraći elementi* vidljivo je da su Tetriminosi definirani kao 2D matrice. Prikazane su matrice za prva 4 elementa. Ćelije koje sadrže vrijednost će biti popunjene bojom određenom za taj element. Ćelije u koje je upisana nula, neće biti ispunjene. Time se dobije oblik željenog elementa. Npr. Element „O“ je polje sa dva polja koje imaju sve vrijednosti postavljene na 2. To znači da će se prilikom crtanja tog elementa, on crtati po cijeloj

dimenziji ove matrice čime se dobije Tetris oblik kockice. Oblik „J“ će se pak ispunjavati na mjestima s brojem 3, dok preostali elementi njegove matrice neće biti obojani jer im je vrijednost postavljena na nulu. Na isti način, definira se i preostalih 5 Tetris elemenata.

```
document.addEventListener("keydown", event =>{
  if(event.keyCode === 37){
    playerMove(-1);
  } else if(event.keyCode === 38){
    playerRotate(1);
  } else if(event.keyCode === 39){
    playerMove(1);
  } else if(event.keyCode === 40){
    playerDrop();
  } else if(event.keyCode === 81){
    playerRotate(-1);
  } else if(event.keyCode === 87){
    playerRotate(1);
  }
});
```

Slika 44 - Upravljanje elementima

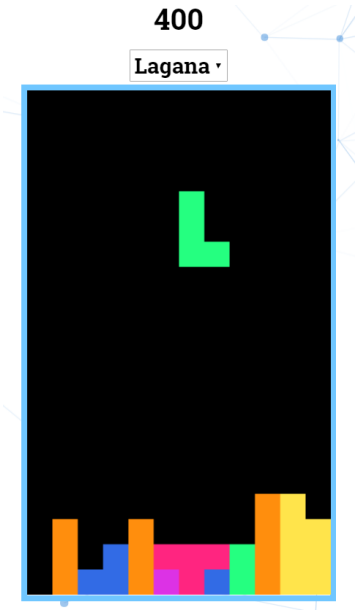
Upravljanje elementima riješeno je na način da se na svaku od korištenih tipki postavi slušač klika. Da bi program prepoznao koju tipku je korisnik kliknuo, koristi se keyCode atribut. Svaka tipka tipkovnice ima svoj ključ. 37, 38, 39, 40 su ključevi za tipke strelica. Klikom na bilo koju od njih, pokreće se odgovarajuća funkcija. Ako se klikne tipka u lijevo, pokreće se playerMove(-1), ako se klikne tipka u desno, pokreće se playerMove(1). Isto tako i za rotaciju (strelica gore/Q/W) i pomicanje u dolje (strelicom na dolje).

```
function playerDrop(){
  player.pos.y++;
  if(collide(arena, player)){
    player.pos.y--;;
    merge(arena, player);
    playerReset();
    arenaSweep();
    updateScore();
  }
  dropCounter = 0;
}
```

Slika 45 - PlayerDrop funkcija

PlayerDrop funkcija pomiče igrača za jedan korak u dolje (svake sekunde inicijalno, tj. brže ako se igra na većoj težini). Zatim provjerava s collide funkcijom da li je došlo do dodira s drugim elementom ili dnom igraćeg polja. Ukoliko je, resetira se pozicija igrača i matrice,

čisti se redak (ili cijela matrica ukoliko je dodir s vrhom igraćeg polja tj. igra završena). I rezultat igrača se ažurira (uvećava za 10 ili resetira na nulu ako je došlo do završetka igre). Ako je igrač izgubio igru (Tetriminos je dotaknuo vrh igraće matrice), otvara mu se modal prozor sa porukom o završetku igre i ispisom rezultata koji je postigao. Ako klikne na gumb „Igraj ponovno“, igraća matrica se čisti, pozicija igrača se resetira na početnu, resetira se rezultat i igra započinje ispočetka. Ukoliko klikne na gumb zatvaranja modal prozora ili „Izlaz“ gumb, izlazi se iz Tetris igre i vraća se u ekran odabira igre.



Slika 46 - Tetris igra

9.6. Stranica odabira memory igre

Klikom na gumb za igranje memory igre prelazi se u na ovaj ekran:



Slika 47 - Glavni dio stranice odabira memory igre

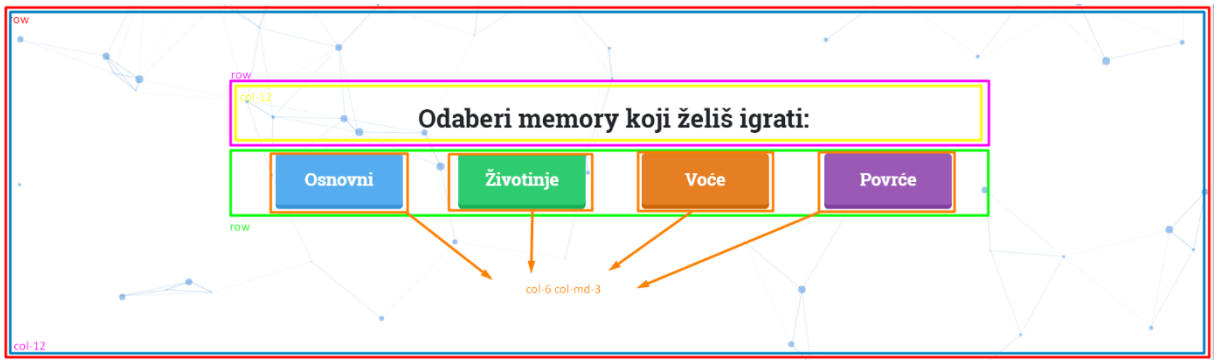
Korisnik bira jedan od 4 moguća modela memory igre.

Osnovna verzija memory-a rađena je prema freeCodeCamp početničkom Javascript tutorialu (https://www.youtube.com/watch?v=ZniVgo8U7ek).

Na osnovnu verziju, nadodane su dodatne 3 varijante igre u kojima igraći prostor sadrži veći broj karata i različite sličice parova. Dodatno, napravljen je i modal prozor koji se otvara pri završetku igre. Modal prozor je grafički element koji je „zamjena“ glavnom prozoru aplikacije. Kad se modalni prozor otvori, glavni prozor se onemogućuje, ali ostaje vidljiv ispod modal prozora. Korisnik mora ugasiti modal prozor (odabirom neke od opcija na modal prozoru ili gašenjem modal prozora) prije nego što se nastavi izvođenje glavnog prozora. Prva verzija igre je osnovna (najmanje igraće područje) i ima svoj grafički stil i memory elemente. Ostala tri memory stila su veća (broj igraćih memory parova je veći) te se svaki razlikuje i dizajnom i memory elementima, tj. temama (životinje, voće, povrće). Klikom na bilo koji od njih prelazi se u željenu igru.



Slika 48 - Razlomljena mreža ekrana za odabir memory igre

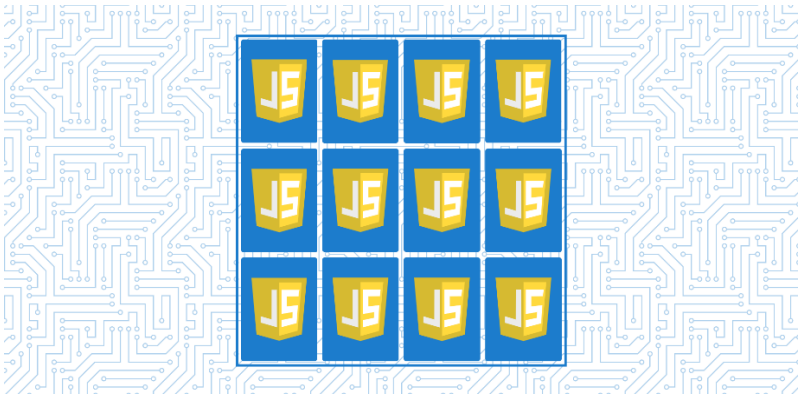


Slika 49 - Mreža ekrana za izbor memory igre

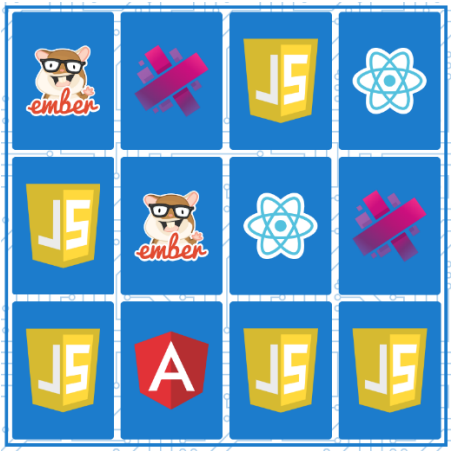
9.7. Memory igra

Memory je poznat kao igra uparivanja. On zahtjeva koncentraciju i dobru memoriju za pobjedu. Memory kao igra pokriva koncept evaluacije računalnog razmišljanja. Zadaci evaluacije sadrže određene uvjete koji moraju biti evaluirani (definirani kao točni ili netočni). Uvjeti i evaluacija uvjeta su bitni jer se učenikove odluke temelje na njima.

Iako je igra dizajnirana da dva igrača igraju jedan protiv drugoga, također ju je moguće igrati samostalno. Cilj igre je sakupiti najveći broj parova (ako se igra u paru), ili pronaći sve parove (ukoliko se igra samostalno). Memory je postavljen kao mreža ili tablica memory kartica kod kojih svaka karta ima odgovarajući par. Sve kartice prvotno su okrenute i vidljiva im je poleđina (koja je ista kod svih). Karte su položene na podlogu "licem" prema dolje, tako da su sve "slike" karata skrivene.



Slika 50 - Memory "osnovni" način igre



Slika 51 - Igra osnovnog memory-a

```
  <link rel="stylesheet" href="memoryStyle.css">
</head>

<body>
  <section class="memory-game">
    <div class="memory-card" data-framework= "aurelia">
      <img class="front-face" src="assets/img/aurelia.svg" alt="Aurelia"/>
      <img class="back-face" src="assets/img/js-badge.svg" alt="JSBadge"/>
    </div>
    <div class="memory-card" data-framework= "aurelia">
      <img class="front-face" src="assets/img/aurelia.svg" alt="Aurelia"/>
      <img class="back-face" src="assets/img/js-badge.svg" alt="JSBadge"/>
    </div>
    <div class="memory-card" data-framework= "vue">
      <img class="front-face" src="assets/img/vue.svg" alt="Vue"/>
      <img class="back-face" src="assets/img/js-badge.svg" alt="JSBadge"/>
    </div>
    <div class="memory-card" data-framework= "vue">
      <img class="front-face" src="assets/img/vue.svg" alt="Vue"/>
      <img class="back-face" src="assets/img/js-badge.svg" alt="JSBadge"/>
    </div>
    <div class="memory-card" data-framework= "react">
      <img class="front-face" src="assets/img/react.svg" alt="React"/>
      <img class="back-face" src="assets/img/js-badge.svg" alt="JSBadge"/>
    </div>
    <div class="memory-card" data-framework= "react">
      <img class="front-face" src="assets/img/react.svg" alt="React"/>
```

Slika 52 - Memory HTML

Igraće polje započinje definicijom HTML-a. Sve karte igre smještene su unutar sekcije klase „memory-game“. Svaka karta je smještena unutar div elementa klase „memory-card“ sa data-framework atributom imena. Kao što je poznato, karte imaju prednju i zadnju stranu. One su pozicionirane u dva img elementa jedna iznad druge, tj. Back-face slika je prva vidljiva, dok se prednja strana slike „skriva“ i vidi se tek nakon što kliknemo na kartu i ona se rotira. Osim stilskih i transformacijskih CSS stilova, najbitniji je stil flip klase zajedno sa vidljivosti prednje strane karte. Flip klasa okreće kartu za 180 stupnjeva, dok ju vidljivost prednje strane postavljena na hidden skriva sve dok korisnik ne klikne na nju.

```
// close icon in modal
let closeicon = document.querySelector(".close");

// declare modal
let modal = document.getElementById("popup1")

const cards = document.querySelectorAll(".memory-card");

let hasFlippedCard = false;
let lockBoard = false;
let firstCard, secondCard;
let counter = 0;
```

Slika 53 - Memory deklaracija globalnih varijabli

U Javascript kodu, prvo se deklariraju potrebne globalne varijable. Prve tri se iz DOM-a selektiraju uz pomoć upravitelja događaja. Selektiraju se sve kartice, te elementi modal prozora koji će biti potrebni kasnije. Ostalih pet je deklarirano kao normalna varijabla.

```
(function shuffle(){
  cards.forEach(card =>{
    let randomPos = Math.floor(Math.random() * 12);
    card.style.order = randomPos;
  });
})();

cards.forEach(card => card.addEventListener("click", flipCard));
```

Slika 54 - Memory početna funkcija

Funkcijom shuffle se započinje program. Ova funkcija se automatski izvodi prilikom njenog deklariranja (što se Javascriptu naznači zatvaranjem funkcije u zagrade i pozivanje iste na kraju s zagradama bez argumenta). Ova funkcija svakoj karti dodjeljuje jedan slučajno odabran broj do 12, na temelju kojih se karte pozicioniraju tj. promiješaju. Nakon postavljanja polja, postavlja se event listener na svaku kartu koji „sluša“ korisnički klik na jednu od karata da dalje pokrene funkciju flipCard.

```
function flipCard(){
  if(lockBoard) return;
  if(this === firstCard) return;

  this.classList.toggle("flip");

  if(!hasFlippedCard){
    //first click
    hasFlippedCard = true;
    firstCard = this;

    return;
  }
  //second card
  hasFlippedCard = false;
  secondCard = this;

  checkForMatch();
  congratulations();
}
```

Slika 55 - Flip funkcija

FlipCard funkcija započinje njenom deklaracijom. Prva provjera provjerava da li je lockBoard varijabla postojana, tj. boolean istinirta, ako je, prekida se izvođenje funkcije. Ova provjera je bitna kod zaustavljanja otvaranja novih karata ukoliko je korisnik u pola animacije otvaranja drugog para. This operator druge provjere vraća element koji je pokrenuo funkciju, odnosno, div element kartice. Ako je on jedan prvoj karti, izvođenje se prekida. Bitan dio flip

funkcije je dodavanje „flip“ klase this elementu (div elementu kartice memory polja). Time se kartica rotira i pokazuje sličicu koju skriva. Toggle funkcija si postavlja pitanje. Da li postoji definirana klasa flip nad div elementom? Ako ne postoji, dodaj je, a ako postoji, ukloni je.

Provjera if(!hasFlippedCard) provjerava da li je varijabla hasFlippedCard false, tj. ako je, onda korsnik otvara prvu kartu od para. Ako program prođe uvjet, varijabla hasFlippedCard se postavlja na istinitost i s this pridružuje se div element kojeg je korisnik kliknuo varijabli firstCard (prva kartica od dvije). Nakon if provjere, hasFlippedCard se vraća na false a div element druge karte se sprema u secondCard varijablu. Ovaj dio je ubiti else dio if naredbe. Nakon prolaska kroz navedene linije, pozivaju i pokreću se funkcije checkForMatch i congratulations.

```
function checkForMatch(){

  let isMatch = firstCard.dataset.framework === secondCard.dataset.framework;

  isMatch ? disableCards() : unflipCards();
}

function disableCards() {
  firstCard.removeEventListener("click", flipCard);
  secondCard.removeEventListener("click", flipCard);
  counter += 1;
}

function unflipCards() {
  lockBoard = true;
  setTimeout(() => {
    firstCard.classList.remove("flip");
    secondCard.classList.remove("flip");

  resetBoard();
  }, 900);
}

function resetBoard(){
  [hasFlippedCar, lockBoard] = [false, false];
  [firstCard, secondCard] = [null, null];

}
```

Slika 56 - Provjera parova, isključivanje karata, resetiranje runde i resetiranje polja

Prva funkcija *Slika 56 - Provjera parova, isključivanje karata, resetiranje runde i resetiranje polja* je checkForMatch. Ona deklarira varijablu isMatch koja se postavlja na boolean vrijednost uvjeta. Uvjet provjerava prethodno spomenute data-framework atribute definirane u HTML dijelu igre. Ukoliko obje karte imaju isti data-framework atribut, poziva se funkcija disableCards. Ako oni nisu isti, poziva se unflipCards funkcija.

Skraćeni način pisanja if provjere naziva se kondicionalni (ternary) operator, te se ovaj način pisanja često koristi kao kratica koja krati i čini kod preglednijim. Ako dođe do izvršenja disableCards funkcije, znači da je isti par karata pronađen, te se sa njih uklanja slušač klik

događaja. Čime se one blokiraju i ostaju otvorene jer su pronađene. Dodatno, brojač se povećava za 1 (bitno kod provjere broja svih pronađenih parova). Ukoliko otvorene karte nisu iste, unflipCards naredba će im ukloniti flip klase (pa će se karte okrenuti). Preventivno se lockBoard varijabla postavlja na istinu (da igrač ne može otvarati nove karte dok se trenutne nisu zatvorile), te se poziva resetBoard funkcija koja resetira igraće polje pa korisnik može krenuti u novu rundu otvaranja novog para karata.

```
function congratulations(){

    if (counter == 6){

        modal.classList.add("show");

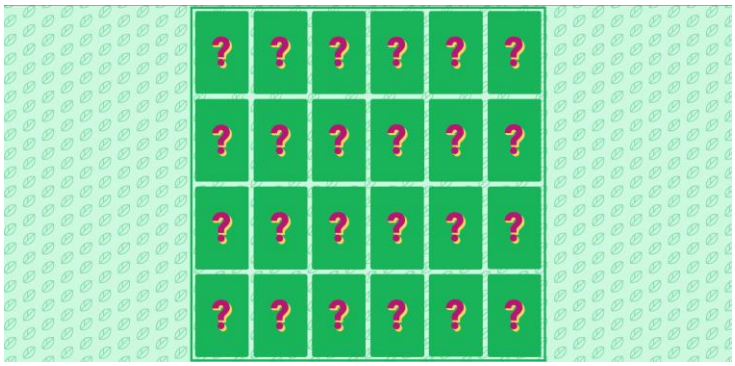

        closeModal();
    };
}


// @description close icon on modal
function closeModal(){
    closeicon.addEventListener("click", function(e){
        modal.classList.remove("show");
        shuffle();
    });
}
```

Slika 57 - Funkcije modal prozora

Slika 57 - Funkcije modal prozora prikazuje funkcije modal prozora. Congratulations funkcija provjerava da li je brojač karata jednak broju 6 (prva memory igra ima ukupan broj parova 6, dok ostale 12). Ako je to slučaj, modalu se postavlja klasa na show, pa ga korisnik može vidjeti. Dodatno pokreće funkciju closeModal koja „sluša“ na korisnikov klik na klik u prostor van modala. Ako se to desi, modalu se miče show klasa i miješaju se karte.

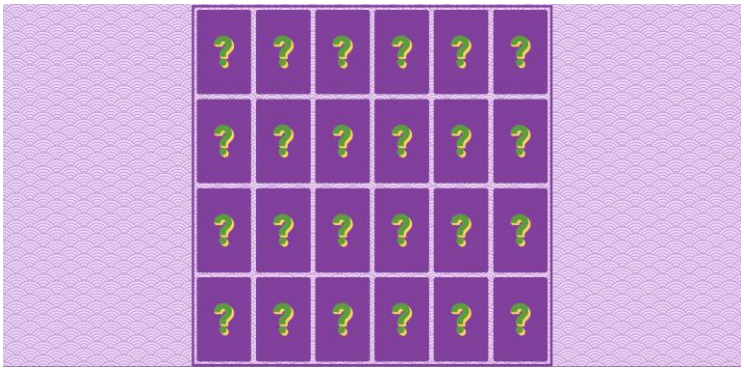
Dodatno, svaka memory igra ima generiranu pozadinsku sliku body elementa. Pozadinske slike su birane da na neki način odgovaraju sadržaju memory kartica. Također, boja linija pozadine je ista kao i kartice, dok je prostor između svijetlije nijanse. Uzorci sa specifičnim bojama su generirani kao link kojeg se uključuje u CSS datoteku u body element. Stranica za generaciju je: http://www.heropatterns.com/



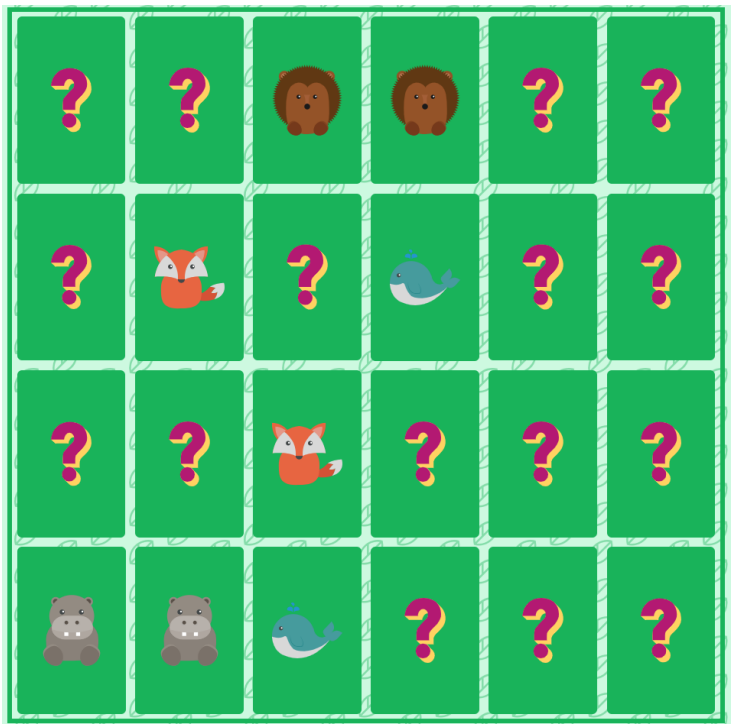
Slika 58 - Memory životinje



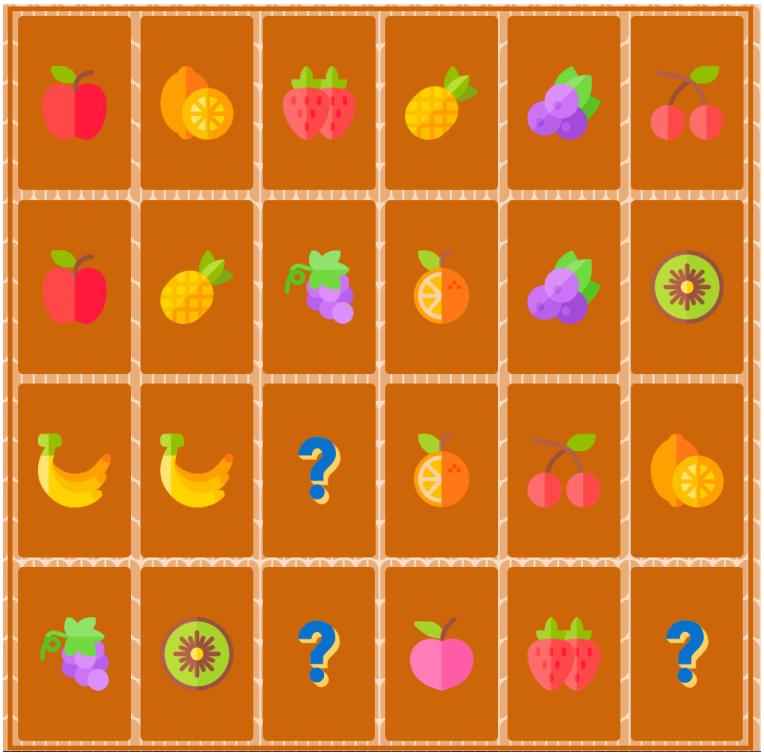
Slika 59 - Memory voće



Slika 60 - Memory povrće



Slika 61 - Memory životinje igra

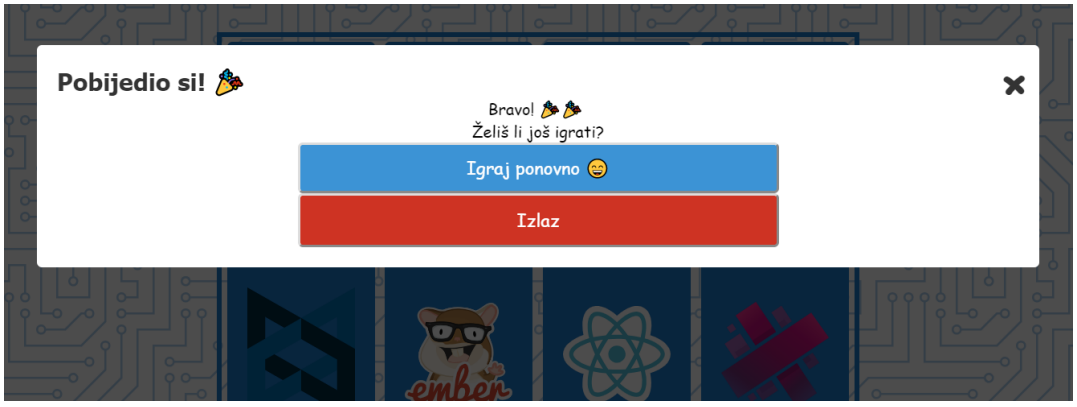


Slika 62 - Memory voće igra



Slika 63 - Memory povrće igra

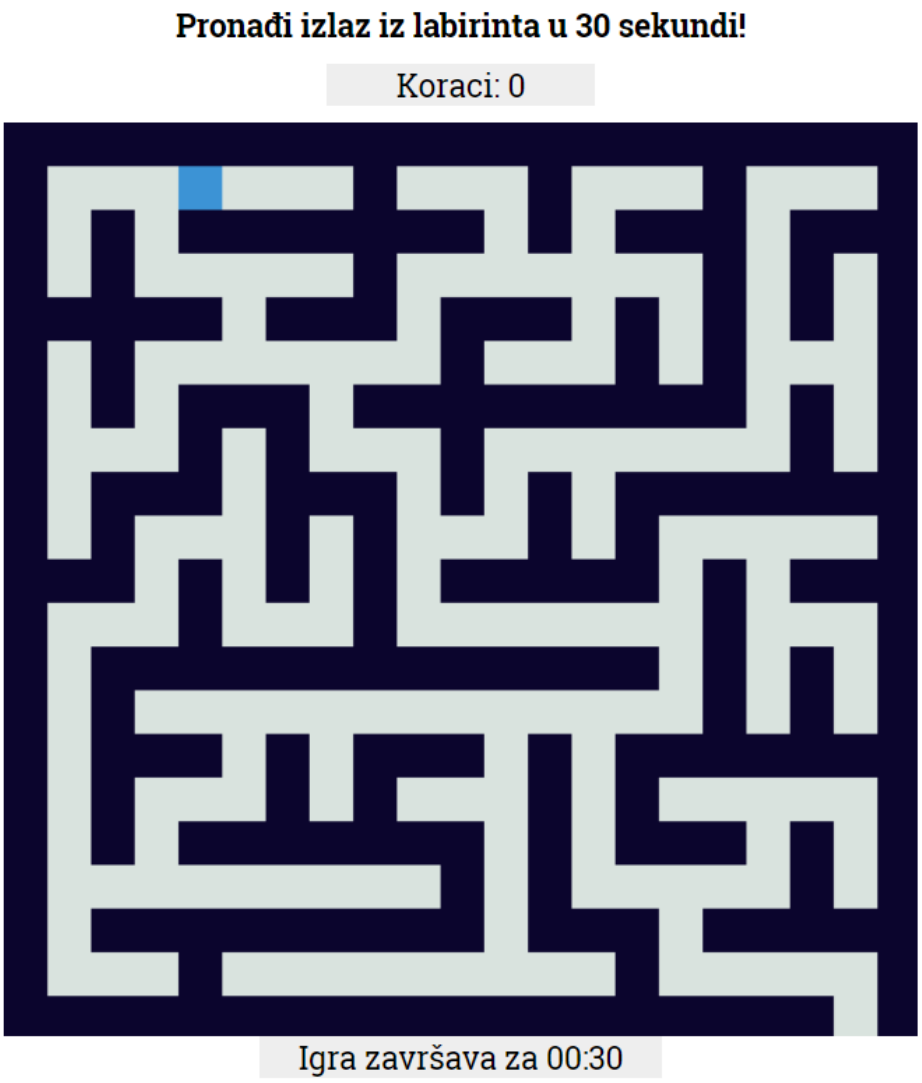
Runda započinje klikom na jednu od kartica, animacija pokreće otvaranje iste i korisniku se prikazuje slika kartice. Nadalje, izabire se nova kartica za okretanje. Ukoliko su obje kartice iste, one se zaključavaju i nova runda odabira novog para karata započinje. Ukoliko kartice nisu identične, one se obje zatvaraju i nova runda odabira započinje. Cilj igre je u što kraćem roku pronaći parove na cijelom igraćem polju. Kad se taj cilj i ostvari, korisniku s otvara modal prozor. Ako korisnik odluči izaći iz igre (klikom na crveni gumb izlaz), vraća se na ekran izbora memory igre. A ako klikne na plavi gumb igraj ponovno, igra se ponovno pokreće. Naravno, kao i u pravoj memory igri, karte se izmiješaju i time je svako pokretanje igre drugačije.



Slika 64 - Modal prozor memory igre

9.8. Labirint igra

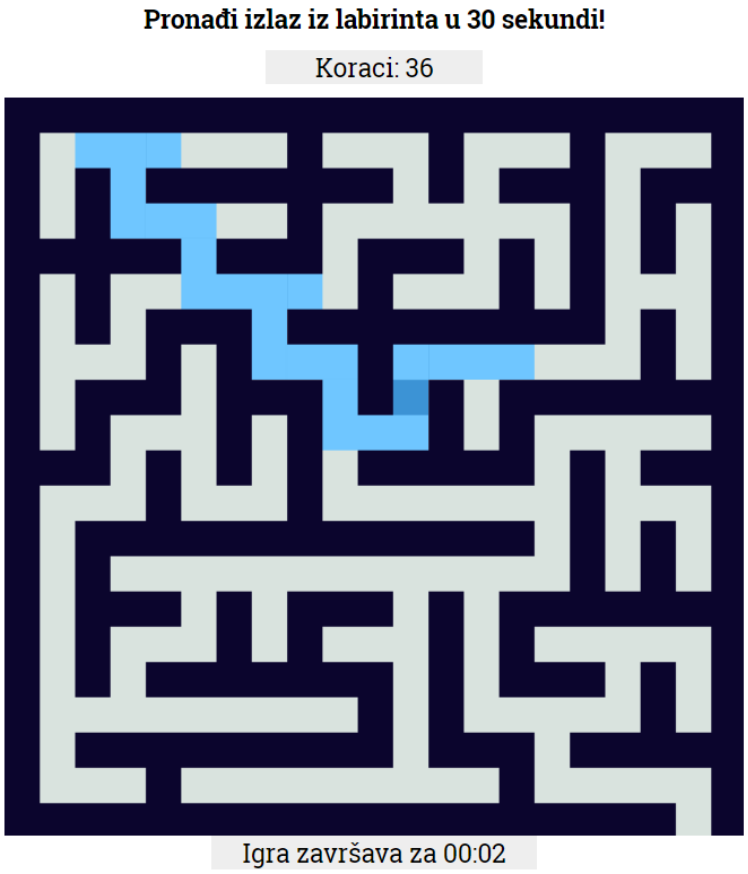
Igra labirinta pokriva koncept Algoritama računalnog razmišljanja. Zadatak se rješava metodom korak po korak, tj., određenim pravilima se dolazi do rješenja. Identifikacijom uzoraka u zadatku, učenici dolaze do koraka koji ih dovode do rješenja zadatka. Na početku igre generira se labirint, te se igrač (plava kockica) pozicionira unutar njega. Cilj igre je igrača dovesti do izlaza labirinta unutar 30 sekundi.



Slika 65 - Labirint igra

Pomoću tipki sa strelicama određuje se gdje će se igrač pomaknuti. Pritiskom tipke sa strelicama ulijevo/udesno/gore/dolje, igrač se pomiče u odgovarajućem smjeru.

Prilikom pomicanja igrača unutar labirinta, iza njega ostaje plavi trag koji označava put koji je igrač prešao.



Slika 66 - Labirint igra, zajedno sa prikazom prijeđenog puta

Ako igrač ne uspije naći izlaz do isteka vremena, igra završava i nudi mogućnost ponovnog pokretanja igre (otvara mu se modal prozor).



Slika 67 - Završni modal prozor

Ukoliko igrač nađe cilj unutar vremenskog ograničenja, igrač je uspješno obavio zadatak. Ponovnim pokretanjem igre (bilo nakon izgubljene/pobijeđene runde), novogenerirani labirint je različit. Odnosno, svakim pokretanjem igre nasumično se generira labirint, čime nijedna runda nije ista. Osim glavnog igraćeg prostora stranice, stranica sadrži zaglavlje koje je identično kao i kod ostalih.

```
<div class="col-12">
<div id="maze">
    <p style="text-align: center;margin-bottom: 10px; font-family: RobotoBold; font-size: 18px;">Pronađi izlaz iz labirinta u 30 sekundi!</p>
    <div id="c" style="margin-left:auto; margin-right:auto;margin-bottom: 10px;text-align: center;width: 10%;font-size: large"></div>

    <canvas id="canvas" width="523" height="523" style="margin-left:auto; margin-right:auto">
    Vaš preglednik ne podržava HTML5 Canvas.
</canvas>
    <div id="timerel" style="margin-left:auto; margin-right:auto;margin-top:0px;text-align: center;width: 15%;font-size: large"></div>

</div>

<!-- The Modal -->
<div id="myModal" class="modal">

    <!-- Modal content -->
    <div class="modal-content">
        <div class="modal-header">
            <span class="close">&times;</span>
            <h2 class="gamehead"></h2>
        </div>
        <div class="modal-footer">
            <h2 id="demo" onmouseover="" style="cursor:pointer;">Zaigraj ponovno?</h2>
        </div>
    </div>
```

Slika 68 - Labirint HTML

Glavni dio HTML stranice sadrži paragraf sa uputnom rečenicom, brojač koraka, vremenski brojač, canvas s samom igrom i skriveni modal prozor koji se otvara uspješnim pronalaskom izlaska iz labirinta ili neuspješnom rundom (istek vremena za izlaz).

```
function startTimer(duration, display) {
    var start = Date.now(),
        diff,
        minutes,
        seconds;

    function timer() {
        if(playing) {
            diff = duration - (((Date.now() - start) / 1000) | 0);
            minutes = (diff / 60) | 0;
            seconds = (diff % 60) | 0;
            minutes = minutes < 10 ? "0" + minutes : minutes;
            seconds = seconds < 10 ? "0" + seconds : seconds;
            display.textContent = "Igra završava za " + minutes + ":" + seconds;

            if (diff <= 0) {
                display.textContent = "Završena igra";
                start = Date.now() + 1000;
                playing = false
                modelfungo();
            }
        }
    };
    timer();
    setInterval(timer,1000)
```

Slika 69 - Vremenska funkcija

SetTimer funkcija očitava trenutno vrijeme te preko timer funkcije odbrojava minutu. Ako diff varijabla padne ispod nule, isteklo je 60 sekundi čime se pokreće modelfungo funkcija koja otvara modal prozor za izgubljenu rundu.

```
window.onload = function () {
    halfminutes = 30;
    x = document.querySelector("#timerel");
    startTimer(halfminutes,x)
}
playing = true
window.addEventListener('keydown',doKeyDown,true);
```

Slika 70 - Početna funkcija i slušač klika

Početna funkcija koja se pokrene prilikom otvaranja stranice je prikazana na *Slika 70 - Početna funkcija i slušač klika*. Vrijeme se postavlja na 30 sekundi, dohvaća se element brojača preko ključa timerel, te se postavlja timer elementa na prethodno definiranih 30 sekundi i funkcija pokreće odbrojavanje. Boolean varijabla playing postavlja se na istinitost i inicijalizira se slušač klika koji pokreće doKeyDown funkciju.

```
function doKeyDown(evt)
{
    var handled = false;
    if (playing) {
        switch (evt.keyCode) {
            case 38:  /* Up arrow was pressed */
                m.moveup("canvas");
                handled = true
                break;
            case 87:  /* Up arrow was pressed */
                m.moveup("canvas");
                handled = true
                break;
            case 40 :  /* Down arrow was pressed */
                m.movedown("canvas");
                handled = true
                break;
            case 83 :  /* Down arrow was pressed */
                m.movedown("canvas");
                handled = true
                break;
            case 37:  /* Left arrow was pressed */
                m.moveleft("canvas");
                handled = true
                break;
```

Slika 71 - Slušači klika

Slično kao i kod Tetrisa, kliknuta tipka se prepoznaje preko njene keyCode vrijednosti, tj ključa. Na temelju kliknute tipke, pokreće se odgovarajuća funkcija. M element preko kojeg se pristupa funkcijama je objekt labirinta. Njega se deklarira, te se pozivaju glavne funkcije programa globalno:

```
m = new maze(10 , 10);
m.init();
m.add_edges();
m.gen_maze();
m.draw_canvas("canvas");
function drawMoves() {
    document.getElementById("c").innerHTML = "Koraci: "+ m.getMoves()
}
// drawMoves();
setInterval(drawMoves, 100);
```

Slika 72 - Globalna deklaracija

Kreira se nova instanca objekta labirinta. Nakon njegove kreacije se pozivaju funkcije: init, add_edges, gen_maze, draw_canvas i postavljanje intervala koje poziva funkciju drawMoves. DrawMoves dohvaća element c u kojeg preko getMoves funkcije upisuje trenutni broj prijeđenih koraka u labirintu. Svaki klik i pomicanje labirintom se broji kao jedan korak. Init funkcija definira osnovnu veličinu igraćeg prostora prema x i y veličinama dodanim iz inicijalizacije konstruktora new maze. Zatim se uz pomoć add_edges funkcije dodaju vanjski rubovi igraćeg prostora. Gen_maze funkcija randomizira i generira puteve i zidove unutar labirinta, te njen rezultat u konačnici crta draw_canvas funkcija.

```
this.checkPos = function (id) {
    for (var i = 0; i < 2 * this.N + 1; i++) {
        for (var j = 0; j < 2 * this.M + 1; j++) {
            if (this.Board[i][j] == '&') {
                // console.log(i,j)
                return [i,j]
            }
        }
    }
}
```

Slika 73 - Provjera pozicije

Slika 73 - Provjera pozicije prikazuje funkciju provjere pozicije. Ona se poziva i provjerava nakon svakog pomaka igrača. Bilo to moveup, movedown, moveleft ili moveright funkcija. Uz pomoć checkPos se dolazi do X i Y koordinata igrača u 2D prostoru canvasa. Dobivena pozicija se uspoređuje sa izlaznom pozicijom preko funkcije checker. S obzirom da je izlaz iz labirinta uvijek na istoj poziciji, vrijednost trenutne pozicije se uspoređuje sa fiksnom pozicijom. Ako je igrač došao do cilja, otvara mu se modal prozor. A ako nije stigao do cilja unutar vremenskog ograničenja, otvara mu se modal prozor s informacijom o neuspjehu.

10. Zaključak

Cilj ovog diplomskog rada bila je izrada odabranih didaktičkih igara vezanih uz GBL područje. Glavna svrha igara (time i cijele web stranice) je njihovo korištenje u edukacijskim svrhama kod učenika mlađe dobi s ciljem poticanja razvoja algoritamskog načina razmišljanja. Algoritamski način razmišljanja neophodan je kod savladavanja programiranja, a njegov razvoj kod djece se ponajviše postiže s raznim didaktičkim igrama. Učenje im na ovaj način postaje zabavno, te u kombinaciji sa tradicionalnim metodama učenja, osigurava da uče efikasno.

Implementacija je izvršena korištenjem osnovnih web tehnologija, ili tzv. Klijentske tehnologije (HTML, CSS i Javascript). Glavna struktura stranice je rađena u HTML5 strukturnom jeziku. CSS je korišten za stilski izgled stranice, zajedno u kombinaciji sa Bootstrap 4 okvirom kojim se osigurala responzivnost cijelog sjedišta. Izrađene su 3 interaktivne igre (Tetris, Memory, Labirint), te svaka od njih pokriva jedan od 5 koncepata algoritamskog razmišljanja. Što se implementacije igara tiče, ona je izvedena uz pomoć Javascript skriptnog jezika. Sami okviri igre smješteni su unutar HTML-ovog canvas grafičkog okvira za crtanje i manipulaciju 2D objekata.

Korištenjem navedenih alata, u ovom radu je prikazana njihova međusobna interakcija, koja je dovela do izrade didaktičkih igara smještenih unutar interaktivnog web sučelja. Ovime se dolazi do zaključka da se kombinacijom ovih temeljnih tehnologijama web-a, može izraditi bilo kakva web stranica/aplikacija. Svaka od njih neophodna je za bilo kakav sadržaj Interneta, te bez njih, bi Internet vjerojatno bio znatno drugačiji od Interneta kakvog danas znamo.

11. Popis slika

Slika 1 - Game Based Learning ilustracija 2
Slika 2 - Slojevita struktura Web-a 3
Slika 3 - Osnovna "trojka" svake web stranice 4
Slika 4 - HTML, CSS i JavaScript logotipi 5
Slika 5 - Osnovna struktura HTML dokumenta 6
Slika 6 - Struktura CSS pravila 7
Slika 7 - Izvođenje stranice 10
Slika 8 - Developer alati unutar web preglednika (Google Chrome) 11
Slika 9 - Stil plavog gumba 11
Slika 10 - Javascript konzola 12
Slika 11 - Javascript debugger 13
Slika 12 - Interno uključivanje Javascript kôda 14
Slika 13 - Javascript komentari 16
Slika 14 - Responzivnost web stranice na različitim uređajima 17
Slika 15 - Medijski upiti i njihovo djelovanje na izgled stranice 18
Slika 16 - Bootstrap mreža i prikaz njezinih 12 stupaca u 5 redaka 19
Slika 17 - Mockup početne stranice 21
Slika 18 - Skica početne stranice 21
Slika 19 - Veliki logo početne stranice 22
Slika 20 - Navigacijski gumbi početne stranice 22
Slika 21 - Podnožje stranice 23
Slika 22 - Zaglavlje stranice 23
Slika 23 - Općeniti opis projekta 24
Slika 24 - Grid korištenih tehnologija i slika 24
Slika 25 - Sudionici projekta 24
Slika 26 - Bootstrap grid sekcije sudionika 25
Slika 27 - Zaglavlje uputa + Tetris opis 26
Slika 28 - Memory opis 26
Slika 29 - Opis Labirinta + podnožje 26
Slika 30 - Bootstrap grid uputa 27
Slika 31 - Zaglavlje stranice selekcije igara 27
Slika 32 - Glavni dio stranice za selekciju igara 28
Slika 33 - Bootstrap grid stranice za selekciju igre 28
Slika 34 - Službeni Tetris logo 29
Slika 35 - 7 elemenata (Tetriminosa) Tetris igre 29
Slika 36 - Tetris HTML 30
Slika 37 - Dohvaćanje canvas elementa 30
Slika 38 - Globalne varijable 31
Slika 39 - Update funkcija 31
Slika 40 - Vremenske varijable 32
Slika 41 - Funkcije crtanja 32
Slika 42 – Igraća matrica 33
Slika 43 - igraći elementi 33
Slika 44 - Upravljanje elementima 34
Slika 45 - PlayerDrop funkcija 34
Slika 46 - Tetris igra 35

Slika 47 - Glavni dio stranice odabira memory igre 35
Slika 48 - Razlomljena mreža ekrana za odabir memory igre 36
Slika 49 - Mreža ekrana za izbor memory igre 36
Slika 50 - Memory "osnovni" način igre 37
Slika 51 - Igra osnovnog memory-a 37
Slika 52 - Memory HTML 38
Slika 53 - Memory deklaracija globalnih varijabli 38
Slika 54 - Memory početna funkcija 39
Slika 55 - Flip funkcija 39
Slika 56 - Provjera parova, isključivanje karata, resetiranje runde i resetiranje polja 40
Slika 57 - Funkcije modal prozora 41
Slika 58 - Memory životinje 42
Slika 59 - Memory voće 42
Slika 60 - Memory povrće 42
Slika 61 - Memory životinje igra 43
Slika 62 - Memory voće igra 43
Slika 63 - Memory povrće igra 44
Slika 64 - Modal prozor memory igre 44
Slika 65 - Labirint igra 45
Slika 66 - Labirint igra, zajedno sa prikazom prijeđenog puta 46
Slika 67 - Završni modal prozor 46
Slika 68 - Labirint HTML 47
Slika 69 - Vremenska funkcija 47
Slika 70 - Početna funkcija i slušač klika 48
Slika 71 - Slušači klika 48
Slika 72 - Globalna deklaracija 49
Slika 73 - Provjera pozicije 49

12. Literatura

[1] »Educational game,« 27. 03. 2018. [Mrežno]. Available: https://en.wikipedia.org/wiki/Educational_game#Game-based_learning. [Pokušaj pristupa 27. 08. 2018.].

[2] E. Team, »What is GBL (Game-Based-Learning),« 23. 04. 2013. [Mrežno]. Available: http://edtechreview.in/dictionary/298-what-is-game-based-learning. [Pokušaj pristupa 27. 08. 2018.].

[3] G. Cahill, »Why Game-Based Learning?,« [Mrežno]. Available: http://thelearningcounsel.com/article/why-game-based-learning. [Pokušaj pristupa 27. 08. 2018.].

[4] A. Kucheriavy, »Which Technology Is Best for My Website?,« [Mrežno]. Available: https://www.intechnic.com/blog/which-technology-is-right-for-my-website/. [Pokušaj pristupa 28. 08. 2018.].

[5] »The difference between HTML, CSS and JAVASCRIPT,« [Mrežno]. Available: https://www.getsmarter.com/career-advice/industry-advice/difference-html-css-javascript. [Pokušaj pristupa 28. 08. 2018.].

[6] J. Collins, »The Basics of Web Technologies,« 13. 07. 2001. [Mrežno]. Available: http://www.alphadevx.com/a/7-The-Basics-of-Web-Technologies. [Pokušaj pristupa 30. 08. 2016.].

[7] »HTML,« 16. 06. 2018. [Mrežno]. Available: https://hr.wikipedia.org/wiki/HTML. [Pokušaj pristupa 30. 08. 2018.].

[8] R. Shannon, »What is HTML?,« 21. 08. 2012. [Mrežno]. Available: https://www.yourhtmlsource.com/starthere/whatishtml.html. [Pokušaj pristupa 30. 08. 2018.].

[9] »HTML5,« [Mrežno]. Available: https://en.wikipedia.org/wiki/HTML5. [Pokušaj pristupa 30. 08. 2018.].

[10] »Cascading Style Sheets,« 30. 08. 2018. [Mrežno]. Available: https://en.wikipedia.org/wiki/Cascading_Style_Sheets. [Pokušaj pristupa 30. 08. 2018.].

[11] J. Wong, »Learn to style HTML using CSS,« 30. 08. 2018. [Mrežno]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS. [Pokušaj pristupa 30. 08. 2018.].

[12] »JavaScript,« 30. 08. 2018. [Mrežno]. Available: https://en.wikipedia.org/wiki/JavaScript. [Pokušaj pristupa 30. 08. 2018.].

[13] w. d. MDN, »What is Javascript?,« [Mrežno]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Pokušaj pristupa 1. 12. 2018].

[14] M. web, »What are browser developer tools?,« [Mrežno]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_are_browser_developer_tools. [Pokušaj pristupa 1. 12. 2018.].

[15] »w3Schools - Javascript events,« [Mrežno]. Available: https://www.w3schools.com/js/js_events.asp. [Pokušaj pristupa 1. 12. 2018.].

[16] »Wikipedia - JavaScript syntax,« 7. 11. 2018.. [Mrežno]. Available: https://en.wikipedia.org/wiki/JavaScript_syntax. [Pokušaj pristupa 1. 12. 2018].

[17] Wikipedia, »Bootstrap (front-end framework),« 14. 11. 2018. [Mrežno]. Available: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework). [Pokušaj pristupa 20. 11. 2018.].

[18] »Responsive web design,« 4. 11. 2018.. [Mrežno]. Available: https://en.wikipedia.org/wiki/Responsive_web_design. [Pokušaj pristupa 20. 11. 2018.].

[19] N. Pettit, »Beginner's Guide to Responsive Web Design,« [Mrežno]. Available: https://blog.teamtreehouse.com/beginners-guide-to-responsive-web-design. [Pokušaj pristupa 20. 11. 2018.].

[20] »CSS,« [Mrežno]. Available: https://getbootstrap.com/docs/3.3/css/. [Pokušaj pristupa 21. 11. 2018.].

[21] »Bootstrap grid system,« [Mrežno]. Available: https://www.w3schools.com/bootstrap/bootstrap_grid_system.asp. [Pokušaj pristupa 21. 11. 2018.].