JavaScript u multimedijskom prikazu znanja

Dobrota, Bruno

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2018./2019.

Bruno Dobrota

JavaScript u multimedijskom prikazu znanja

Završni rad

Mentor: prof.dr.sc. Tomislava Lauc

Zagreb, ožujak 2019.

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Bruno Dubravka

(potpis)

Zahvaljujem profesorici Tomislavi Lauc na punoj podršci od prvog dana kada je nastala ideja za izradom projekta te kasnije kroz svu pomoć na završnom radu.

Sadržaj

1. Uvod

Čovjek kao radoznalo biće teži k vlastitom usavršavanju i razvitku. U tom dugotrajnom procesu učenja, naša zajednica je osigurala trajan napredak i mogućnost konstantnog guranja vlastitih granica i otkrivanje novog. Kao glavna komponenta unutar evolucijskog napretka čovječanstva leži znanje. Njena bit je glavni pokretač svog razvoja koji karakterizira današnji svijet, njezino shvaćanje i korištenje jedina je konkluzija koja je dovela čovjeka u moderan svijet koji omogućuje lepezu mogućnosti. U informatičkom razvijenom okruženu, programski jezici postali su presedan za obavljanja različitih zadataka koji uključuju tehnološka znanja i suvremene računalne tehnologije. Jedan od popularnih programskih jezika jest JavaScript. U ovom radu će se ukratko objasniti osnovni postupci uporabe web tehnologija, HTML-a i JavaScripta, na primjeru izrade multimedijskog tutorijala. Multimedijski tutorijal sadrži elemente multimedija u svrhu multimedijskog prikaza znanja. Tutorijal prikazan u ovom radu temelji se na dva osnovna elementa, a to su tekst i slika. Pri tome je tekst u pisanom, a slika u statičnom obliku. U radu je opisano na koji način moderne tehnologije poput programskog jezika JavaScripta mogu biti korištene za izradu multimedijskog tutorijala, tj. primjeri uporabe JavaScripta objašnjeni su prikazom izrade projekta koji je za cilj imao prikazati znanje iz područja fotografije. Budući da fotografija kao jedno od područja unutar umjetnosti zauzima veliki opseg tema, tutorijal obuhvaća tek osnovne komponente i bitne karakteristike unutar tema boje i kompozicije fotografije. Oboje je izuzetno bitno za stjecanje općenitog osnovnog znanja o fotografiji te njihovom primjenom korisnik ima mogućnost brzog savladavanja cjelokupnog područja. Cilj rada je prezentirati kako programski jezik JavaScript i HTML element *canvas* mogu biti korišteni za izradu multimedijskog tutorijala.

2. HTML

HTML, akronim za engleski naziv *Hypertext Markup Language*, jest osnovni jezik za stvaranje Web stranica. Prema Shannon (2012), *Hypertext* je način na koji se krećete na webu - klikom na poseban dio koji se zove hiperveza i koji vas vodi na novi sadržaj – „*hiper“* znači da možete otići na bilo koje mjesto na Internetu kad god želite klikom na poveznice. „*Markup* označava što HTML oznake rade s tekstom unutar njih, obilježavajući ga kao određenu vrstu teksta*“* (Shannon, 2012; vlastiti prijevod). Flannagan (1997) navodi da HTML može napraviti dokument interaktivnim, putem hipertekstualnih veza koje povezuju određeni dokument s drugim dokumentima. HTML kao svojevrsni sustav za oblikovanje informacija i sadržaja na Webu omogućuje pohranu pojedinačne stranice kao zasebnog dokumenta, s ekstenzijom *.html*. „Preglednik koji se koristi čita datoteku i prevodi tekst u korisniku prezentiran oblik, nadajući se da će prikazati stranicu kako je autor htio“ (Shannon, 2012; vlastiti prijevod). Možemo reći da je HTML jezgra svake Web stranice budući da se ostale komponente i jezici poput CSS-a, PHP-a i JavaScripta povezuju s HTML-om. Ovaj najzastupljeniji jezik na Internetu omogućuje korištenje i prikazivanje različitih elemenata poput teksta, slike, videa, tablica i slično. Prvo spominjanje HTML-a i Interneta bilo je 1991. godine od strane Tim Berners-Leeja, dok je prva verzija HTML-a objavljena 1993. godine. Verzija koja obilježava današnjicu, te se pretežno koristi, zove se HTML5. Nastala je 2014. godine, a trenutna verzija HTML5.2 preporučena je od strane World Wide Web Consortium (*W3C*) 2017. godine. „HTML5 je sljedeća generacija HTML-a koja podržava nove značajke potrebne modernim web aplikacijama“ (Harmadi, Kaluža i Liverić, 2014, str. 199).

„HTML dokument sastoji se od sadržaja dokumenta i oznaka koje definiraju strukturu i izgled dokumenta“ (Musciano i Kennedy, 1997; vlastiti prijevod). Osnovni kostur stranice (Slika1), sadrži nekoliko glavnih oznaka svakog HTML koda a to su: *<html> </html>, <head> </head>* i *<body> </body>*. Oznake kao osnovna građa pomažu pri stvaranju strukture jedinica sadržaja, te se one jedino vizualno mogu vidjeti u samoj Web stranici i nisu vidljive korisniku. Zaglavlje (eng. *head*) je mjesto gdje dajete HTML dokumentu naslov i gdje naznačite druge parametre koje preglednik može koristiti prilikom prikazivanja dokumenta, dok je tijelo (eng. *body*) mjesto gdje stavljate stvarni sadržaj HTML dokumenta. Primjer prikazuje kako je unutar elementa *<body>* zadano prikazivanje glavnog naslova, tj. upotrebom *<h1>* elementa na stranici se prikazuje: "Ovo je naslov". Zatim se pomoću elementa *<p>* omogućuje prikaz odlomka teksta "Ovo je paragraf."

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>

        <h1>Ovo je naslov.</h1>
        <p>Ovo je paragraf.</p>
    </body>
</html>
```

Slika 1: Osnovni kostur dokumenta u HTML-u

3. JavaScript

„JavaScript je programski jezik pomoću kojeg se mogu ostvariti velike promjene u HTML dokumentu, uključujući animaciju, interaktivnost i dinamičnost vizualnih efekata“ (McFarland, 2014, str. xiii; vlastiti prijevod). JavaScript se izvršava na klijentskoj strani i izvorni kod obrađuje klijentski web-preglednik. Povijest ovog programskog jezika seže do kraja prošlog stoljeća, odnosno godine 1995., i Brendana Eicha koji je također kreator i vlasnik *Netscape*-a. „JavaScript je izvorno bio zamišljen kao skriptno sučelje između Web stranice učitane u klijentski preglednik (u to vrijeme *Netscape Navigator*) i aplikacije na poslužitelju“ (Powers, 2013, str. xi). Također, Haverbeke (2014) ističe da je JavaScript omogućio stvaranje modernih Web aplikacija koje omogućuju izravno komuniciranje bez ponovnog učitavanja stranice za svaku akciju koju korisnik poduzme. „Na primjer, funkcija JavaScripta može provjeriti web obrazac prije slanja kako bi provjerila jesu li ispunjena sva potrebna polja“ (Christensson, 2014; vlastiti prijevod). „JavaScript kôd može proizvesti poruku o pogrešci prije nego se bilo koja informacija doista prenese na poslužitelj“ (Christensson, 2014; vlastiti prijevod). JavaScript se često koristi za izvođenje operacija koje bi inače opteretile poslužitelj (eng. *server*). „Integriran je u okruženje za pregledavanje, što znači da može dobiti informacije o pregledniku i HTML stranici te izmijeniti te informacije, mijenjajući time način na koji se stvari prikazuju na zaslonu“ (Shannon, 2012; vlastiti prijevod).

Razlog velikog interesa programera za primjenu ovog programskog jezika te njegove sve veće upotrebe jest u tome što je vodeći za domenu Interneta. Kao i ostali skriptni jezici, brz je, budući da interpreter prevodi kod red po red. „Provođenje JavaScript naredbi započinje odmah, budući da za svoje izvođenje JavaScript ne zahtijeva vezu s poslužiteljem“ (Harmadi, Kaluža i Liverić, 2014, str. 200). Napretkom računalnih tehnologija, aplikacije „sele“ na Internet te postaju dio *online* platforme gdje su svima dostupne. U takvom okruženju, JavaScript se najčešće pokazuje kao odlično rješenje za određeni *framework* poput *Angualra*. *Angular* je platforma, odnosno programersko sučelje koje pomaže za programiranje u HTML i JavaScriptu. „Jedan od razloga zašto je tako popularan jest upravo razmjerno lako dodavanje skripti Web stranici“ (Powers, 2013, str. 1). Crockford (2008) smatra da je JavaScript izuzetno važan kao jezik Web preglednika. Radi se o objektno orijentiranom jeziku koji veliki dio svojih funkcija zasniva na događajima (eng. *events*), omogućujući dinamičnost u interakciji između korisnika i računala.

JavaScript usko surađuje s HTML-om, i nadovezuje na kostur stranice i njezine elemente na dva načina:

1. unutar HTML dokumenta kao element <script> </script>
2. izvan HTML dokumenta, odnosno kao samostalni *.js* dokument koji se povezuje s osnovnim HTML dokumentom tako da se unutar zaglavlja doda element:
   < link rel="stylesheet" href="naziv_datoteke.js" >

3.1 Primjeri sintakse JavaScripta

Sintaksa programskog jezika je skup određenih pravila koja se moraju poštovati kako bi se program mogao pravilo izvršiti, a važno početno pravilo JavaScripta jest razlikovanje malih i velikih slova, što često može biti uzrok greške u programu. Za svaki od tipova objekata, naredbi, operatora ili nekih drugih komponenti JavaScripta postoje određena pravila sintakse, a njihova funkcija i točnost, naravno, utječe na kvalitetu i ispravnost programa.

Nekoliko je osnovnih elemenata koje omogućuju stvaranje prvih redova koda, a jedna od njih je varijabla. „Varijabla je imenovana referenca na podatak, a podatak može biti niz znakova (eng. *string*), broj (eng. *number*) ili logička vrijednost (*true* ili *false*), kao i referenca na funkciju, polje ili drugi objekt“ (Powers, 2013, str. 19). „JavaScript varijable imaju identifikator, doseg i specifičan tip podataka“ (Powers, 2013, str. 20). Važno je uvažavati pravila sintakse jezika.

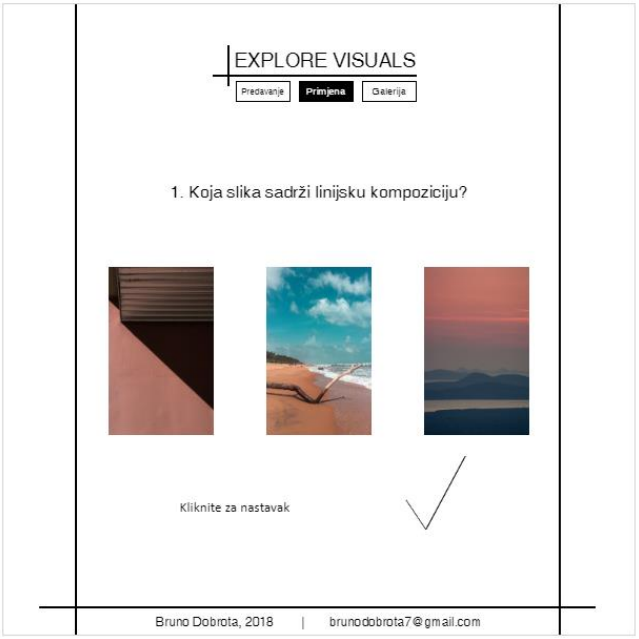
Nadalje, operatori kao i u svakom programskom jeziku, označavaju određenu operaciju i povezuju jedan ili više operanda u jedan izraz. Glavna podjela unutar ove skupine jest na aritmetičke, unarne, usporedne i logičke operatore. Aritmetičke operatore čine zbrajanje (+), oduzimanje (-), množenje (*),i dijeljenje (/). Oni se nazivaju binarnim operatorima jer zahtijevaju dva operanda. Unarni operatori se razlikuju od aritmetičkih po tome što je njihova primjena na samo jedan operand, a ima ih tri: povećanje vrijednosti (++), smanjivanje vrijednosti (--) te negativna vrijednost (-). Usporedni operandi uspoređuju te vraćaju *true* ili *false*. To su „==“ koji predstavlja jednakost, „<“ i „>“ koji predstavljaju „manje od“ i „veće od“ , te manje ili jednako„<=“ i veće ili jednako„>=“ te različito. „!=“. Logičkih operanda ima tri te su podijeljeni na logičko I (eng. *and*) koje se označava simbolom „&&“ i vraća vrijednost *true* samo ako su svi uvjeti zadovoljeni, logičko ILI (eng. *or*) koje se označava simbolom „||“ a vraća vrijednost *true* ako je jedan od uvjeta točan, te logičko NE (eng. *not*) koje se označava „!“ te je dio i unarnih operanda a označava inverziju (Flannagan, 1997).

Operatori i operanadi kombiniraju se u izraze koji su dozvoljeni u jeziku, a svaki izraz iza kojeg slijedi točka–zarez predstavlja naredbu. Naredbe su iduća bitna stavka. Naredba ili izjava (eng. *statement*) u JavaScriptu govori pregledniku što treba raditi. Prema McFarlandu (2014), izjave su bazična programska jedinica, najčešće prezentirana kao jedan korak unutar JavaScript koda. Crockford (2008) radi podjelu naredbi na kondicionalne - uvjetne (*if* i *switch*), ponavljajuće (*for*, *while* i *do*) te disruptivne, odnosno remetilačke (*break*, *return* i *throw*) izjave.

Izjava *if* koristi se za provjeru zadovoljenja uvjeta, pa se kod koji slijedi iza uvjeta zadanog ovom naredbom može izvršiti samo ako je zadovoljen taj uvjet. Kao što je niže prikazano (Slika 2), izjava ili naredba *if* zahtijeva da se uvjet zadan u *( )* (uglatim) zagradama mora prvo ispuniti (odnosno biti *True*) da bi se naredbe koje slijede u vitičastim zagradama mogle izvršiti. Također, ako se uvjet ne ispuni i želimo da se u tom slučaju određena radnja dogodi, dodaje se naredba *else*. Njena funkcija je izvršavanje radnje ako uvjet uz *if* naredbu nije ispunjen (*False)*, te se postavljaju druge mogućnosti i naredbe za izvršavanje. Na primjerima iz projekta (Slika 2, Slika 3 i Slika 4) prikazana je uporaba *if* naredbe: ako je uvjet ispunjen (*True),* iscrtava se slika (kvačica) koja prikazuje da je odgovor točan, te se točnim odgovorima pripisuje dodatan bod vrijednosti, a ako uvjet nije ispunjen, uz uporabu *else* naredbe prikazuje se slika netočnog odgovora (križić) te se istome pridružuje dodatna bodovna vrijednost.

```
118          if(a==CorrectAnswer){
119            context.drawImage(quizbg, 70,650,100,100,380,430,75,70);
120            rightanswers++;
121          }
122          else{
123            context.drawImage(quizbg, 260,650,100,100,380,450,75,70);
124            wronganswers++;
125          }
```

Slika 2: *if- else* naredba



Slika 3: Primjer kvačice

Slika 4: Primjer križića

Nadalje, jedna od najčešćih struktura podataka JavaScripta, također opisana na primjeru projekta, jesu polja ili nizovi vrijednosti. Prema Flannaganu (1997), polje (eng. *array*) je tip podataka koji sadrži numerirane dijelove podataka. Numerirani dio je element polja, broj dodijeljen elementu je njegov indeks. Također, *array* odnosno polje se može tumačiti i kao niz ili lista, no to prevladava više u drugim programskim jezicima poput Pythona. Element niza može biti bilo kojeg tipa, a različiti elementi istog niza mogu biti različitih vrsta. Za stvaranje

polja koriste se uglate zagrade []. Slika niže (Slika 5) prikazuje *Array* čiji su članovi pitanja korištena za kviz tutorijala. Svaki element je tip podatka *string,* zapisan u navodnicima ("").

```
43                  var Questions = ["1. Koja slika sadrži linijsku kompoziciju?",
44                                   "2. Koja slika sadrži minimalizam",
45                                   "3. Koja slika sadrži pravilo trećine?",
46                                   "4. Koja slika sadrži simetriju?",
47                                   "5. Koja slika sadrži balans?"];
```

Slika 5*: JavaScript Array*

4. Canvas HTML5

Canvas je HTML5 element koji uz crtanje otvara mnoge dizajnerske mogućnosti za uređivanje sadržaja. „HTML5 *canvas* nudi programerima mogućnost stvaranja animirane grafike u običnim web-preglednicima pomoću uobičajenih alata: HTML i JavaScript“ (Fulton i Fulton, 2011, str. xv; vlastiti prijevod). „HTML5 definira element <*canvas*> kao rastersku površinu za crtanje koja ovisi o rezoluciji izraženoj u pikselima, te se može koristiti za iscrtavanje grafova, grafike igara i različitih slikovnih prikaza“ (Pilgrim, 2010, str. 16). Dvije glavne značajke pri definiranju *canvasa* su njegova visina i širina (*height & width*) i one određuju veličinu površine za crtanje. „Crtaća površina je pravokutnik na stranici po kojem je moguće pomoću JavaScript programskog jezika nacrtati razne oblike“ (Harmadi, Kaluža i Liverić, 2014, str. 199). „*Canvas* je element koji je izvorno podržan unutar web-preglednika i dio je objektnog modela dokumenta (eng. *Document Object Model*, *DOM*)“ (Hickson, 2014 prema Harmadi, Kaluža i Liverić, 2014, str. 199).

Možemo reći da *canvas* omogućuje poboljšanja u dizajnu kao i ostavljanje dobrog utiska na korisnika. „HTML5 *canvas* je bitmapirano područje na ekranu u neposrednom načinu rada kojim se može manipulirati pomoću JavaScripta“ (Fulton i Fulton, 2011, str. 1; vlastiti prijevod). Budući da se zasniva na pikselima, neki stručnjaci ga smatraju najadekvatnijim rješenjem za kompleksno crtanje po određenoj Web stranici. „Svaki objekt koji se nalazi unutar *canvas* elementa strogo je definiran brojem i pozicijom piksela, što znači da je *canvas* ovisan o rezoluciji“ (Harmadi, Kaluža i Liverić, 2014, str. 200).

Početno stvaranje *canvasa* i dobivanje osnovnog oblika omeđenog pravokutnikom prikazat ćemo primjerom sa stranice *w3schools* tutorijala za *canvas.* Atributom *style* definiramo određene značajke stila elementa. Točnije, kao što je niže prikazano (Slika 6 i Slika 7), moguće je odabrati rub *canvasa* debljine jednog piksela te sive boje. „Posljednji i vjerojatno najvažniji korak je dati našem elementu Atributi za širinu (eng. *width*) i visinu (eng. *height*) definiraju veličinu prostora za crtanje.

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

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
</canvas>

<script>
</script>

</body>
</html>
```

Slika 6: Osnovno postavljanje *canvasa* u HTML-u (preuzeto s: https://www.w3schools.com/graphics/tryit.asp?filename=trycanvas_filltext)

Slika 7: Prikaz iscrtanog canvasa na stranici (preuzeto s: https://www.w3schools.com/graphics/tryit.asp?filename=trycanvas_empty)

„Crtaća površina je pravokutnik na stranici po kojem pomoću JavaScripta možete nacrtati što god želite“ (Pilgrim, 2010, str. 16). „<canvas> stvara fiksnu površinu za crtanje koja izlaže jedan ili više „kontekst“ za ostvarivanje prikaza (eng. rendering contexts) u koji se izdaju JavaScript naredbe za crtanje i manipuliranje prikazanog sadržaja“ (Ivančić, 2012, str. 3). „Kao što smo upravo spomenuli, jedna od prvih stvari koju trebamo učiniti kada stavimo *canvas* na HTML5 stranicu je testirati da vidimo je li cijela stranica učitana i jesu li svi HTML elementi prisutni prije nego počnemo s izvođenjem bilo kakvih operacija“ (Fulton i Fulton, 2011, str. 7; vlastiti prijevod).

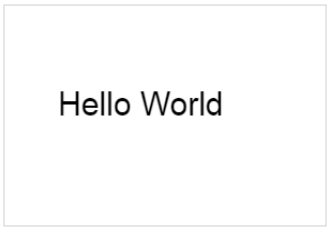
```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="300" height="200" style="border:1px solid #d3d3d3;">
</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",50,100);
</script>

</body>
</html>
```

Slika 8: Prikaz naredbi za ispis teksta Hello World (preuzeto s: https://www.w3schools.com/graphics/tryit.asp?filename=trycanvas_filltext)



Slika 9: Prikaz na stranici prema kodu sa slike 8 (preuzeto s: https://www.w3schools.com/graphics/tryit.asp?filename=trycanvas_filltext)

„Za svaki element (platno) moguće je koristiti "kontekst", te izdati JavaScript naredbe za crtanje svega što želite“ (Sucan, 2009; vlastiti prijevod).

Na primjer:

Izjava: *var c = document.getElementById (“myCanvas”);*

omogućuje preuzimanje elementa *“myCanvas”* te njegovu daljnju primjenu.

Izjava: *var ctx = c.getContext (“2d”);*

omogućuje stvaranje bilo kakvog 2D sadržaja na istom „*canvasu*“.

U ovom radu opisuje se uporaba 2D konteksta. Nadalje, moguća je uporaba 3D konteksta. WebGL (Web Graphics Library) jest JavaScript API za interaktivnu 3D i 2D grafiku.

„*Canvas* element ima DOM (eng. *Document Object Model*) metodu *getContext* koja se koristi za dobivanje konteksta i funkcija za crtanje *getContext*() prima jedan parametar – tip konteksta“ (Ivančić, 2012, str. 4). „*The Document Object Model* predstavlja sve objekte na HTML stranici“ (Fulton i Fulton, 2011, str. 5; vlastiti prijevod). Prethodno, metoda *getElementById* omogućuje preuzimanje elementa, odnosno čvora, u ovom slučaju *canvasa.*

4.1 Crtanje na *Canvasu*

Kao glavna karakteristika i prednost HTML5 *canvasa* svakako je mogućnost kvalitetnog crtanja. Ono omogućuje kreativnost koju programer ima i želi je primijeniti na Web stranicu, a raznim crtačkim metodama to je moguće postići. Također, crtanjem teksta, na *canvasu* je u potpunosti moguće „pisanje“ te se ono također konstantno primjenjuje. Crtanje osnovnih oblika se može podijeliti na stvaranje linija te oblika poput pravokutnika ili kruga.

4.1.1 Crtanje linija

Crtanje možemo započeti jednostavno, povlačenjem linija određenih duljina u određenom smjeru. U primjeru (Slika 10), metodom *moveTo(x koordinata, y koordinata)* započinjemo našu liniju, a njezina dva parametra unutar zagrade odnose se na položaj početne točke linije u prostoru *canvasa*. Za dobivanje linije u potpunosti, treba nam krajnja točka koja također ima atribute svojeg položaja unutar prostora *canvasa*, a ona se ostvaruje metodom *lineTo(x koordinata, y koordinata)*. Metodom *stroke()* iscrtava se linija, a opcionalno se može koristiti i metoda *fill()*, kao i kombinacija. U primjeru, u linijama koda od 17 do 22 možemo vidjeti kako su napisane vrijednosti za iscrtavanje dvaju linija, a potom prikaz na stranici (Slika 10). Također, debljinu, vrstu i boju linija, moguće je definirati ovisno o preferencijama dizajna.

```
14              var c = document.getElementById('myCanvas');
15              var ctx = c.getContext('2d');
16
17              ctx.moveTo(165, 265);
18              ctx.lineTo(425, 265);
19              ctx.stroke();
20              ctx.moveTo(185, 210);
21              ctx.lineTo(185, 290);
22              ctx.stroke();
```

Slika 10: Crtanje linija

Slika 11: Prikaz na stranici, dvije linije

4.1.2 Crtanje pravokutnika i kruga

Crtanje oblika moguće je iscrtavanjem linija ili crtanjem pravokutnika kao zasebne cjeline. To je moguće korištenjem metode *rect(x, y, width, height),* kao i *fillRect(x, y, width, height)* koja crta ispunjeni pravokutnik. Nadalje, metoda *strokeRect(x, y, width, height)* crta samo obrub pravokutnika, a metoda *clearRect (x, y, width, height)* briše područje *canvasa* s obzirom na zadanu poziciju, te širinu i visinu. Uporaba *fillRect(x, y, width, height)* metode i svojstva *fillStyle* kojim je određena boja ispune pravokutnika, prikazana je na slikama koje slijede (Slika 12 i Slika 13).

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
```

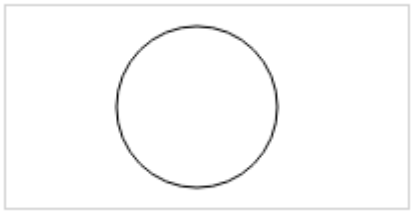
Slika 12: Crtanje pravokutnika na *canvasu* (preuzeto s: https://www.w3schools.com/graphics/tryit.asp?filename=trycanvas_draw)

Slika 13: Prikaz pravokutnika na stranici (preuzeto s: https://www.w3schools.com/graphics/tryit.asp?filename=trycanvas_draw)

Crtanje krugova, kružnica te sličnih oblika, ostvaruje se kao skupina oblika koja se temelji na *arc* metodi. Ista metoda se sastoji od šest parametara: *arc(x, y, radius, start_angle, end_angle, anticlockwise)*. Pozicije *x* i *y* se odnose na razmještaj po platnu s obzirom na središte, *radius* je bitan za određivanje veličine kružnog oblika, *start_angle* označava početni kut dok *end_angle* označava krajnji kut, a parametar *anticlockwise* može poprimiti vrijednosti *True* ili *False*, ovisno ide li se u smjeru kazaljke na satu ili obrnuto. ako vrijednost nije definirana kao u primjeru niže, crta se u smjeru kazaljke na satu. Metoda *beginPath()* koristi se za početak crtanja. Metoda *stroke()* iscrtava definiranu putanju (Slika 14 i Slika 15).

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
```

Slika 14: Crtanje kruga na *canvasu* (preuzeto s: https://www.w3schools.com/graphics/tryit.asp?filename=trycanvas_circle)



Slika 15: Crtež kruga na *canvasu* (preuzeto s: https://www.w3schools.com/graphics/tryit.asp?filename=trycanvas_circle)

4.2 Tekst na *Canvasu*

„2D Canvas za tekst koristi metode fillText(tekst, x koordinata, y koordinata) i strokeText (tekst, x koordinata, y koordinata)“ (Haverbeke, 2014, str. 292; vlastiti prijevod). „Pisanje“ je dakako jednostavnije od crtanja, ali zahtjeva određene druge komponente stila. Na primjer, izjava *ctx.fillText(“*EXPLORE*”*, 300, 252); koristi metodu *fillText* i kao prvi argument upisujemo tekst koji želimo ispisati, nakon toga njegovu poziciju unutar prostora *canvasa*. Također, možemo utjecati na stil i položaj samog teksta. Dodavanjem izjave *ctx.font = “*20pt Helvetica*”;* određujemo veličinu i vrstu fonta, kao što je u primjeru prikazano (Slika 16). Za dodatnu manipulacijom izgleda teksta unutar prostora *canvasa* možemo napisati *ctx.textAlign = “center“;* kako bi centrirali tekst s obzirom na njegovu zadanu poziciju. Primjeri na slikama niže prikazuju tekst koji je na naslovnoj stranici tutorijala. Slika 16 prikazuje kod za tekst koji je prikazan na stranici (Slika 17), ali ujedno i objedinjuje prethodna potpoglavlja vezano za crtanje na *canvasu*.

```
13              //naslov
14              var c = document.getElementById('myCanvas');
15              var ctx = c.getContext('2d');
16              ctx.font = "bold 30pt Helvetica";
17              ctx.fillStyle = "#FFFFF";
18              ctx.textAlign="center";
19              ctx.fillText("EXPLORE",300,252);
20              ctx.font = "20pt Helvetica";
21              ctx.fillText("VISUALS",300,295);
22              ctx.moveTo(165, 265);
23              ctx.lineTo(425, 265);
24              ctx.stroke();
25              ctx.moveTo(185, 210);
26              ctx.lineTo(185, 290);
27              ctx.stroke();
```

Slika 16: „Pisanje“ i crtanje linija na *canvasu*

Slika 17: Naslovna stranica tutorijala

4.3 Slika na *Canvasu*

Vizualno rješenje, poput slike, može u velikoj mjeri doprinijeti dizajnu sadržaja. Dizajneri i programeri često koriste fotografije kao stilski dodatak koji uprizoruje određeni sadržaj. 'Crtanje' slike na platnu odvija se korištenjem metode *drawImage*. Ako želimo sliku podesiti tako da određujemo njezinu širinu i duljinu u prikazu, potrebno je kod metode *drawImage* odrediti parametre – *width* i *height*: *drawImage (image, x, y, width, height)*. Prikazana je uporaba naredbe *var slika = new Image ( );* pri čemu se stvara objekt tipa *Image* i poziv na njega pohranjuje se u varijablu *slika* (Slika 18). Objekti se kreiraju operatorom *new*, u ovom primjeru, pomoću konstruktora *Image()*. Nadalje, potrebno odrediti izvor slike, koristeći svojstvo *src*, pa u našem slučaju imamo izjavu *slika.src = „slika.jpg“;*. Kako bi se slika iscrtala nakon učitavanja, pri otvaranju koristimo metodu *onload*, – *slika.onload = function ( ){...}.* Sve što iduće pišemo, odnosi se na naredbe unutar funkcije te je potrebno da to bude u vitičastim zagradama. U ovom koraku je potrebno nacrtati sliku kako bi mogla biti pozicionirana na poziciju koju želimo te veličine koju želimo, u primjeru naredba: *ctx.drawImage (img, 80, 155, 130, 170);*.

```
143             //slika1
144             var img = new Image();
145             img.src = "slika7.jpg";
146             img.onload = function () {
147                 var c = document.getElementById('myCanvas');
148                 var ctx = c.getContext('2d');
149                 ctx.drawImage(img, 80, 155, 130, 170);
150             }
```

Slika 18: Crtanje slike na *canvasu*

Slika 19: Prikaz slike na stranici

4.4 Kreiranje navigacije uporabom *Canvasa*

Postavljanjem *canvasa* manjih proporcija te pozicioniranjem u određeni dio stranice, kao i uz uporabu metode *onclick* i hiperveza, moguće je ostvariti navigaciju kroz hipermedijski sadržaj. Kako bi omogućili navigaciju uporabom više *canvasa*, potrebno je prvo napraviti više *canvasa*. Slike niže (Slika 20 i Slika 21) prikazuju kod koji omogućuje prikaz izbornika na stranici (Slika 22).

```
12      <canvas id="predavanje" width="50" height="18"
13              style="z-index: 2;position:absolute; left:230px; top:82px; border:1px solid black; background-color: black">
14      </canvas>
15      <canvas id="primjena" width="50" height="18"
16              style="z-index: 3;position:absolute; left:290px; top:82px; border:1px solid black;">
17      </canvas>
18      <canvas id="galerija" width="50" height="18"
19              style="z-index: 4;position:absolute; left:350px; top:82px; border:1px solid black;">
20      </canvas>
```

Slika 20: Postavljanje *canvasa* za navigaciju

```
26              //meni
27              var c = document.getElementById('predavanje');
28              var ctx = c.getContext('2d');
29              ctx.font = "6pt Helvetica";
30              ctx.fillStyle = "white";
31              ctx.textAlign="center";
32              ctx.fillText("Predavanje",25,12);
33
34              var c = document.getElementById('primjena');
35              var ctx = c.getContext('2d');
36              ctx.font = "7pt Helvetica";
37              ctx.fillStyle = "black";
38              ctx.textAlign="center";
39              ctx.fillText("Primjena",25,12);
40
41              var c = document.getElementById('galerija');
42              var ctx = c.getContext('2d');
43              ctx.font = "7pt Helvetica";
44              ctx.fillStyle = "black";
45              ctx.textAlign="center";
46              ctx.fillText("Galerija",25,12);
```

Slika 21: Postavljanje teksta za navigaciju

Slika 22: Izbornik

Potom „*canvase*“ treba učiniti klikabilnim, i napraviti poveznice na druge stranice (Slika 23), o čemu će biti više riječi u narednom poglavlju.

```
78              //funkcija za meni
79              var c = document.getElementById('predavanje');
80              c.onclick =   function() {
81              location.href = "predavanje.html";
82              }
83              var c = document.getElementById('primjena');
84              c.onclick =   function() {
85              location.href = "primjena.html";
86              }
87              var c = document.getElementById('galerija');
88              c.onclick =   function() {
89              location.href = "galerija.html";
90              }
```

Slika 23: Postavljanje funkcija za navigaciju

5. Funkcije, metode i događaji

Prema Powers (2013), funkcije kao bitan dio JavaScript sintakse i logike su fleksibilni objekti koji su sposobni izvršavati određene zadane naredbe. Također, Powers (2013, str. 103) naglašava „funkcionalnost kao bitnu karakteristiku jer funkciju možemo dodijeliti varijabli ili elementu polja ili ju proslijediti kao argument pozivu druge funkcije“. Powers (2013) navodi da unutar samog programskog jezika, izradi funkcija se može pristupiti na tri načina: deklarativni odnosno statički način, dinamički odnosno anonimni način te doslovno. „Najčešći tip funkcije koristi deklarativno/statistički format“... „Ovaj pristup započinje ključnom riječi *function* koju slijedi ime funkcije, zagrade koje sadrže nula ili više argumenata i zatim tijelo funkcije:“ (Powers, 2013, str. 103)

function *imeFunkcije* (parametar 1, parametar 2, ..., parametar n) {
    iskazi funkcije;
}

Za razliku od funkcija koje započinju ključnom riječi *function*, te potom imenom funkcije, anonimna funkcija nema ime (vidi Slika 23). „Ovakav tip funkcije često se naziva anonimnom funkcijom jer ona sama nije izravno deklarirana ili imenovana“ (Powers, 2013, str. 107). „Svaki put kada je pozvana, funkcija se dinamički preoblikuje“ (Powers, 2013, str. 107). „Anonimni izrazi funkcija se nazivaju lambda izrazi (ili kraće lambde) u drugim programskim jezicima“ (Wagner, 2014; vlastiti prijevod). „Sintaksa anonimne funkcije uz upotrebu konstruktora:“ (Powers, 2013, str. 107)

var *imeFunkcije* = new Function („parametar1“, „parametar2“, ... , „parametarn“, „tijelo funkcije“);

„Literali funkcije poznati su i kao izrazi funkcije jer je funkcija napisana kao dio izraza umjesto kao zaseban tip iskaza.“ (Powers, 2013, str. 110). „Oni su nalik na anonimne funkcije po tome što nemaju specifično ime funkcije.“ (Powers, 2013, str. 110). „Osim činjenice da se funkcija dodjeljuje varijabli, literali funkcija nalikuju deklarativnim funkcijama:“ (Powers, 2013, str. 110)

var *imeFunkcije* = function (parametar 1, parametar 2, ..., parametar n) {
    iskazi funkcije;
}

Ono što razlikuje metodu od funkcije jest to što metoda stoji uz varijablu, odnosno objekt, dok je funkcija samostalna. „Događaje *(*eng*. events)* pokreće korisnik u interakciji sa

stranicom, bilo da se radi o klikanju ili pomicanju miša s obzirom na zadano mjesto“ (Shannon, 2012; vlastiti prijevod). U primjeru niže (Slika 24), metoda *onclick* omogućuje se da klikom na određeno područje koje zadamo, izvrši naredba preusmjeravanja na novu stranicu. Pobliže objašnjeno, u ovom primjeru, je na 32. liniji koda definirano područje cijelog *canvasa* tako da bude ‘klikabilno’. To znači da se funkcijom prikazanom u 31. liniji koda izvršava preseljenje pregleda zaslona na lokaciju hiperlinka pomoću svojstva *href* i njemu dodijeljenog imena domene (*location.href = URL)* u 33. liniji koda.

Događaji (eng. *events*) su bitna komponenta unutar JavaScripta. „Događaji se pokreću kad se unutar Web stranice izvede neka aktivnost, primjerice kada se dovrši učitavanje stranice“ (Powers, 2013, str. 145). Također, autorica naglašava da je događaje lako razumjeti te su pridruženi elementima stranice i nisu svojstveni samom jeziku. „*World Wide Web Consortium* (*W3C*) kategorizira događaje prema tri različita područja: događaji korisničkog sučelja (miš, tipkovnica), logički događaji (rezultat procesa) i mutacijski događaji (akcija koja mijenja dokument)“ (Powers, 2013, str. 145).

```
30              //funkcija
31              var c= document.getElementById('myCanvas');
32              c.onclick =   function myFunction() {
33                   location.href = "ime_datoteke.html";
34                   }
```

Slika 24: *Onclick* metoda

Za razliku od metode *onclick* koja se izvršava pri korisnikovom pritisku na određeno klikabilno područje, metoda *onload* se izvršava pri samom učitavanju stranice, odnosno određenog elementa. *Click event* pripada vrsti događaja korisničkog sučelja koji odnose na upotrebu miša, dok *load event* pripada logičkim događajima jer je rezultat procesa. *Load*, kao događaj omogućuje izvršavanje određenog zadatka po učitavanju određenog objekta. U primjeru (Slika 25), iscrtava se slika na *canvasu* uz uporabu metode *onload*.

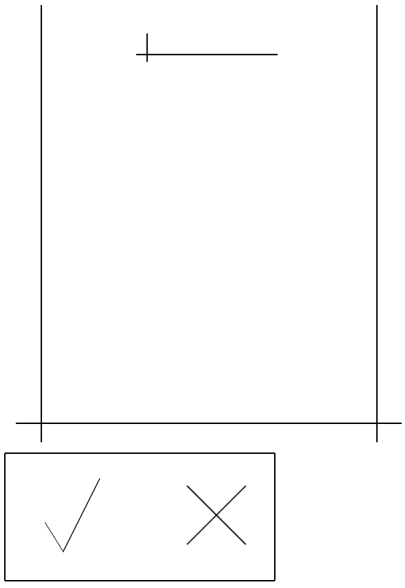
```
51                  //onload
52                  quizbg.onload = function(){
53                    context.drawImage(quizbg, 0, 0);
54                    SetQuestions();
55                  }//quizbg
56                  quizbg.src = "pozadina.png";
```

Slika 25: *Onload* metoda

5.1 Kviz

Projekt sadrži dva kviza, prvi kviz propituje poznavanje boje, dok se drugi odnosni na znanje dobiveno predavanjima o temi kompozicije slike. Prvi se odnosi na odabir između tri ponuđena tekstualna odgovora vezana za boje, dok drugi kviz ima slike kao ponuđene točne i netočne odgovore vezano za temu kompozicije u fotografiji. Oba kviza su napravljena pomoću preuzetog originalnog koda i predloška[1]. U nastavku opisuje kviz sa slikovnim odgovorima. Budući da predložak već sadrži tekstne odgovore kao opcije, dodavanjem kviza sa slikovnim odgovorima proširen je i prilagođen zadani predložak.

Nakon što korisnik jednom odgovori na postavljeno pitanje, upućuje ga se da klikne za nastavak, i kada to učini na *canvasu* se 'briše' postojeće pitanje i njegov sadržaj te se prelazi na novo pitanje. Prema predlošku, za kviz je potrebno pripremiti pozadinsku sliku koja će sadržavati kvačicu i križić van okvira *canvasa* kako ne bi bili vidljiv na platnu, te kako bi se ta dva elementa mogla zasebno izrezati kada je potrebno ilustrirati je li odgovor bio točan ili netočan. Na slici niže (Slika 26) prikazana je pozadina koja sadrži vidljiv dio na *canvasu*, te donji dio koji ima kvačicu i križić koji se ne vide kao pozadina, ali ih je moguće uzimati kao isječke sa slike i za vrijeme korisnikovog odgovaranja na pitanja i iscrtavati na *canvasu*.



Slika 26: Pozadina kviza

[1] Alexander, E. Creating a Multiple Choice Quiz in HTML5 [online]. Dostupno na: http://www.flashbynight.com/tutes/html5quiz/ [25.03.2019.]

Cijeli postupak se pokreće po učitavanju svega na stranicu (metoda *onload* na *14.* liniji koda, Slika 27). Postavljaju se standardne izjave za stvaranje *canvasa* i mogućnost kreiranja njegovog sadržaja, (nalaze se u 16. i 17. liniji koda). Varijabli *quzibg* dodijeljena je pozadinska slika kviza instanciranjem objekta pomoću konstruktora, *new Image( )*. Mogući odgovori prikazuju se kao slike. Varijabla *Question* (linija 24, Slika 27) predstavlja instancu objekta *string*, i odnosi se na postavljanje pitanja. Varijable *Opcija1-3* predstavljaju slikovne odgovore za odgovarajuće pitanje (linija 20-22, Slika 27).

```
13          <script>
14              window.onload = function(){
15
16                  var canvas = document.getElementById("myCanvas");
17                  var context = canvas.getContext("2d");
18
19                  var quizbg = new Image();
20                  var opcija1 = new Image();
21                  var opcija2 = new Image();
22                  var opcija3 = new Image();
23
24                  var Question = new String;
```

Slika 27: Početni kod kviza

Na slici 28 prikazan je nastavak koda. Varijable *mx i my* imaju početnu vrijednost nula, a njihova vrijednost mijenjat će se s obzirom na prepoznavanje koordinata *(x i y)* pri klikanju mišem radi odabira odgovora. Varijabla *CorrectAnswer* koristit će se za upućivanje na točan odgovor - 1, 2 ili 3 koji odgovaraju opciji višestrukog izbora 1, 2 ili 3. Varijabla *QuizFinished* je postavljena na vrijednost *False* te ona služi kako bi se pratio tijek kviza, odnosno ona će se promijeniti u vrijednost *True* kad kviz bude završen. Zatim se nalazi varijabla *lock* koja također ima zadanu *Boolean* vrijednost *False.* Ta će varijabla pomoći da "zaključamo" ponuđene odgovore, tako da korisnik ne može kliknuti na njih više od jednom. Zatim, pomoću varijable *qnumber* pratimo trenutni broj pitanja. Dvije varijable koje se zovu *rightanswers* i *wronganswers* koriste se za pohranjivanje točnih, odnosno netočnih odgovora. Na kraju, *picturepos* (1-4) varijable omogućuju pozicioniranja pitanja i ponuđenih odgovora.

```
var mx=0;
var my=0;
var CorrectAnswer = 0;
var QuizFinished = false;
var lock = false;

var qnumber = 0;
var rightanswers=0;
var wronganswers=0;
var picturepos1=180;
var picturepos2=250;
var picturepos3=250;
var picturepos4=250;
```

Slika 28: Potrebne varijable u početku kviza

Na slici niže (Slika 29) prikazane su linije koda koje sadrže pitanja i odgovore. Varijabla *Questions* je niz koji sadrži pet pitanja koja će se nalaziti u kvizu, dok je varijabla *Options* „niz nizova“, odnosno niz koji sadrži sve ponuđene opcije (nizove) za sva pitanja. Odgovori su ponuđeni kao slike. Također, u komentaru je dodano da se točan odgovor u uvijek nalazi na prvom mjestu što je važno za otkrivanje pozicije točnog odgovora, a bit će objašnjeno u nastavku.

```
var Questions = ["1. Koja slika sadrži linijsku kompoziciju?",
                 "2. Koja slika sadrži minimalizam",
                 "3. Koja slika sadrži pravilo trećine?",
                 "4. Koja slika sadrži simetriju?",
                 "5. Koja slika sadrži balans?"];

var Options = [["IMG_7870.jpg","Slika3.jpg","IMG_5366.jpg"],
["IMG_4508.jpg","IMG_5716.jpg","IMG_4547.jpg"],
["IMG_8205.jpg","IMG_7698.jpg","IMG_7879.jpg"],
["IMG_7699.jpg","IMG_7731.jpg","Slika4.jpg"],
["IMG_7909.jpg","Slika1.jpg","Slika17.jpg"]]; //točan odgovor uvijek na prvom mjestu
```

Slika 29: Nizovi s pitanjima i odgovorima

Sljedeće, potrebno je pripremiti pozadinsku sliku za prikaz pri svakom otvaranju bilo kojeg dijela kviza. Kao što je objašnjeno u poglavlju „Crtanje slike“, koristeći svojstvo *src* i metodu *drawImage* crtamo sliku na *canvasu* nakon što je učitana na stranicu (Slika 30).

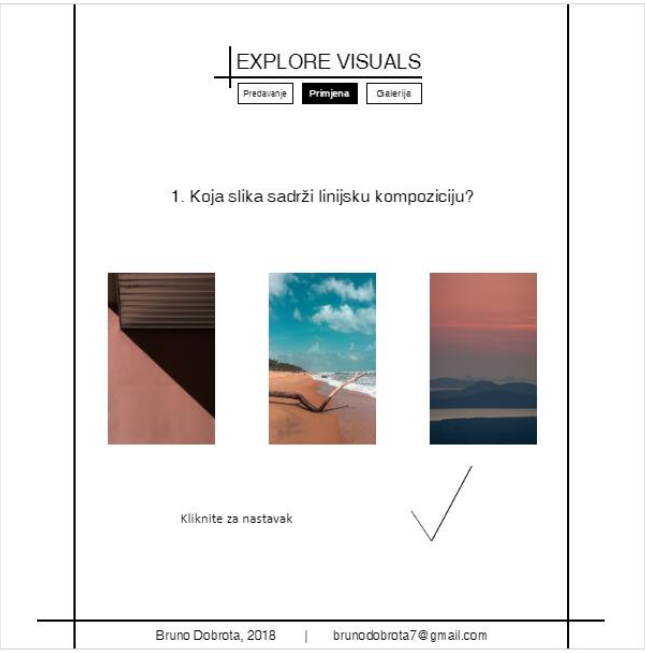
```
quizbg.onload = function(){
  context.drawImage(quizbg, 0, 0);
  SetQuestions();
}
quizbg.src = "pozadina.png";
```

Slika 30: Postavljanje pozadine u kvizu

Slika niže (Slika 31) prikazuje linije koda koje definiraju nasumično postavljanje redoslijeda ponuđenih odgovora za svako novo pitanje. Prvo, odabire se slijedom pitanje iz niza i dodjeljuje varijabli *Question*. Zatim se odabire slučajni broj od 1 do 3 i dodjeljuje varijabli *CorrectAnswer*. Postavit ćemo opcije na takav način da se *CorrectAnswer* odnosi na opciju s točnim odgovorom. Ovisno o slučajnom odabiru varijable, dodjeljujemo vrijednosti varijabli *Option1*, *Option2* i *Option3* varijablama *opcija1*, *opcija2* i *opcija3*. *Options [qnumber] [0]* uvijek sadržava točan odgovor. Dakle, ako je *CorrectAnswer* == 1;, točan odgovor će biti prva opcija, ako je *CorrectAnswer* == 2;, točan odgovor će biti druga opcija, a ako je *CorrectAnswer* == 3;, točan odgovor će biti treća opcija. Funkcija *SetQuestions()* sadrži niz naredbi. Naredbe omogućuju ispis pitanja i opcija koje čine ponuđene odgovore, a sadrže slike i njihov prikaz na stranici (Slika 32).

```
61              SetQuestions = function(){
62
63                  Question=Questions[qnumber];
64                  CorrectAnswer=1+Math.floor(Math.random()*3);
65
66                  if(CorrectAnswer==1){Option1=Options[qnumber][0];Option2=Options[qnumber]
                    [1];Option3=Options[qnumber][2];}
67                  if(CorrectAnswer==2){Option1=Options[qnumber][2];Option2=Options[qnumber]
                    [0];Option3=Options[qnumber][1];}
68                  if(CorrectAnswer==3){Option1=Options[qnumber][1];Option2=Options[qnumber]
                    [2];Option3=Options[qnumber][0];}
69
70                  context.textBaseline = "middle";
71                  context.font = "12pt Helvetica";
72                  context.fillText(Question,300,textpos1);
73
74                  //slike
75                  opcija1.src = Option1;
76                  opcija1.onload = function (){context.drawImage(opcija1,100,textpos2,100,160);}
77
78                  opcija2.src = Option2;
79                  opcija2.onload = function(){context.drawImage(opcija2,250,textpos3,100,160);}
80
81                  opcija3.src = Option3;
82                  opcija3.onload = function(){context.drawImage(opcija3,400,textpos4,100,160);}
83
84              }//SetQuestions
```

Slika 31: Kod za postavljanje pitanja i odgovora u kvizu



Slika 32: Prikaz pitanja i odgovora u kvizu

Slika niže (Slika 33) prikazuje linije koda kojima se omogućuje provjera klikom na ponuđeni odgovor (sliku). Dakle, sljedeći korak je omogućiti korisniku da klikne na odgovore, odnosno na ponuđene slike kako bi dobio povratnu informacije o tome je li odgovor ispravan ili pogrešan (Slika 32). Da bismo to učinili, elementu *canvas* dodajemo osluškivač (eng. *listener)*, odnosno omogućujemo izvođenje naredbe *canvas.addEventListener('click', ProcessClick, false);* Osluškivač događaja odnosi se na klikanje mišem na platnu (*canvasu*). Funkcija *ProcessClick* koristi varijablu *my* za pohranu y-koordinata miša, varijablu te *mx* za pohranu x-koordinata miša, te svojstva *offsetTop* i *offsetLeft* te *pageX* i *pageY* kako bi dobili koordinate u odnosu na platno, a ne u odnosu na cijelu web stranicu.

```
87                  canvas.addEventListener('click',ProcessClick,false);
88
89                  function ProcessClick(ev) {
90
91                  my=ev.y-canvas.offsetTop;
92                  mx=ev.x-canvas.offsetLeft;
93
94                  if(ev.y == undefined && ev.x ==undefined){
95                      my = ev.pageY - canvas.offsetTop;
96                      mx = ev.pageX - canvas.offsetLeft;
97                  }
```

Slika 33: Određivanje koordinata za označavanje odgovora u kvizu

Ako je ispunjen uvjet koji se odnosi na zabilježbu da je korisnik odabrao neki od ponuđenih odgovora, omogućeno je da se ide na novo pitanje te se koristi funkcija *ResetQ()* o kojoj će više biti riječ u sljedećim odlomcima. Pri određivanju područja na koje je korisnik kliknuo, a za svaki mogući odgovor poziva se funkcija *GetFeedback(a)* s parametrom *a* o čemu će također biti riječ u sljedećem odlomku (Slika 34).

```
101             if(lock){
102                 ResetQ();
103             }//if lock
104
105             else{
106
107             if(my<410 && my>240 && mx<200 && mx>100){GetFeedback(1);}
108             if(my<410 && my>240 && mx<350 && mx>250){GetFeedback(2);}
109             if(my<410 && my>240 && mx<500 && mx>395){GetFeedback(3);}
110
111             }//!lock
112
113                 }//ProcessClick
```

Slika 34: Klikanje na ponuđene odgovore u kvizu

Slika niže (Slika 35) opisuje sljedeće: varijabla *a* odnosi se na vrijednosti 1,2 ili 3, ovisno o tome koji je odgovor kliknut. Ako je vrijednost varijable *a* jednaka vrijednosti varijable *CorrectAnswer* onda je odgovor točan, u suprotnom, odgovor je pogrešan. Ako je odgovor točan, želimo s pozadinske slike izrezati kvačicu, te je iscrtati. Ako je odgovor netočan, želimo s pozadinske slike izrezati križić, te ga iscrtati. To možemo učiniti pomoću parametara metode *drawImage*, koji su: *image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight*. Dakle, *sx* je *x* koordinata izvora, *sy* je *y* koordinata izvora. *sHeight* i *sWidth* su visina i širina isječka, koji se zatim ponovno crta prema: *dx* i *dy* (koordinatama odredišta), *dWidth* i *dHeight* (odredišne dimenzije, koje mogu odgovarati izvornim dimenzijama). Izvršava se naredba *rightanswers* ++; kako bi se povećao broj točnih odgovora, odnosno naredba *wronganswers* ++; kako bi se povećao broj netočnih odgovora. Konačno, postavljamo *lock = true;* što sprječava korisnika da ponovno odgovori na isto pitanje. Međutim, ako korisnik ponovno klikne na zaslon, želimo prijeći na sljedeće pitanje koristeći funkciju *ResetQ( )* (Slika 35).

```
116        GetFeedback = function(a){
117
118          if(a==CorrectAnswer){
119            context.drawImage(quizbg, 70,650,100,100,380,430,75,70);
120            rightanswers++;
121            //drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)
122          }
123          else{
124            context.drawImage(quizbg, 260,650,100,100,380,450,75,70);
125            wronganswers++;
126          }
127          lock=true;
128          context.font = "10pt Calibri,Arial";
129          context.fillText("Kliknite za nastavak",220,480);
130        }//get feedback
```

Slika 35: Funkcija dobivanja rezultata odgovora

Na slici niže (Slika 36) opisani su sljedeći postupci. Otključavamo kviz kako bi korisnik mogao odgovoriti na sljedeće pitanje. Funkcija *ResetQ()* omogućuje postavljanje novog pitanja nakon što je korisnik odgovorio. Metoda *clearRect (x, y, width, height);* koristi za brisanje cijelog platna kako bi se prikazao novi sadržaj. Povećava se vrijednost varijable *qnumber* kako bi se pratio redoslijed pitanja koje postavljamo. Ako je vrijednost varijable *qnumber* jednaka vrijednosti *Questions.length* zadnje je pitanje dostignuto. Svojstvo *length* vraća broj elemenata niza, u ovom slučaju niza *Questions*. Nadalje, poziva se funkcija za kraj kviza *EndQuiz().* Ako uvjet nije ispunjen, iscrtava se pozadinska slika kviza i poziva funkcija *SetQuestions( );* radi postavljanja sljedećeg pitanja.

```
133        ResetQ= function(){
134        lock=false;
135        context.clearRect(80,100,430,440);
136        qnumber++;
137        if(qnumber==Questions.length){EndQuiz();}
138        else{
139        context.drawImage(quizbg, 0, 0);
140        SetQuestions();}
141        }
```

Slika 36: Funkcija ResetQ

Uklanjanjem osluškivača (eng. *Listener)* koji je ranije dodan, klikovi na platnu (*canvasu*) više ne bilježe. Posljednja stranica nakon završetka kviza prikazuje rezultate tako da se ukupan broj točnih i ukupan broj netočnih odgovora prikažu na stranici.

```
144        EndQuiz = function(){
145            canvas.removeEventListener('click',ProcessClick,false);
146            //context.drawImage(quizbg, 0,0,550,90,0,0,600,600);
147            context.font = "14pt Calibri,Arial";
148            context.fillText("Završili ste kviz!",300,280);
149            context.font = "10pt Calibri,Arial";
150            context.fillText("Točni odgovori: "+String(rightanswers),300,320);
151            context.fillText("Krivi odgovori: "+String(wronganswers),300,340);
152        }
```

Slika 37: Funkcija EndQuiz i kraj kviza

Slika 38: Završetak kviza

6. Projekt multimedijskog prikaza znanja

6.1 Cilj projekta

Cilj projekta bio je prikazati znanje iz područja fotografije. Budući da fotografija kao jedno od područja unutar umjetnosti zauzima veliki opseg tema i informacija, tutorijal obuhvaća tek osnovne komponente i bitne karakteristike unutar tema boje i kompozicije fotografije. Oboje je izuzetno bitno za stjecanje općenitog osnovnog znanja o fotografiji te njihovom primjenom korisnik ima mogućnost brzog savladavanja cjelokupnog područja.

Izrada tutorijala za multimedijsko učenje zahtjeva određene resurse, alate, metodologiju i znanje. Njihovo organizirano korištenje dovodi do realizacije projekta i mogućnosti njegovog korištenja.

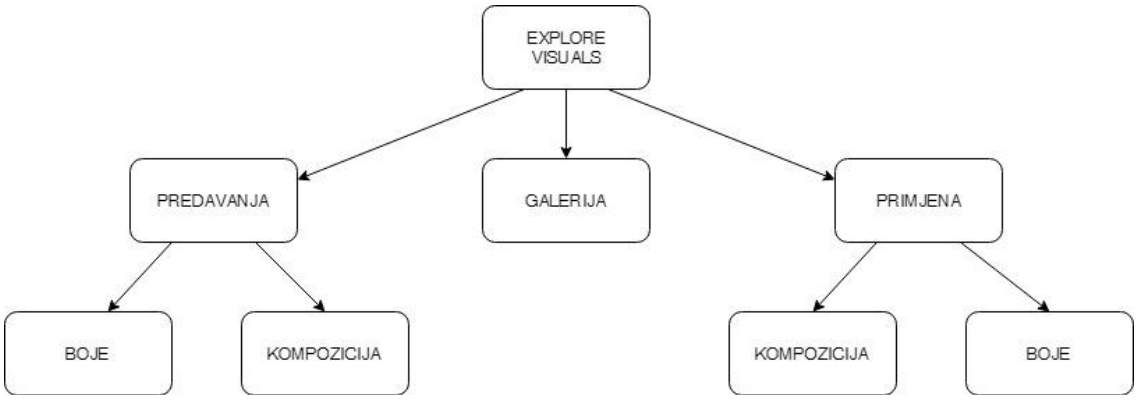
6.2 Alati

Programski alati i rješenja pripadaju skupini neizostavno važnih komponenti za stvaranje projekta. Ovdje se radi o korištenju HTML-a, grafičkog elementa *canvas* te JavaScripta. Kod je realiziran unutar programa otvorenog sustava za upisivanje, uređivanje te provođenja kodova pod nazivom *Brackets*. Ovaj softver, kreiran od strane kompanije *Adobe Systems* koristi se u svrhu *Web developmenta*, odnosno programiranja za potrebe izrade Web stranica i sličnih sadržaja.

6.3 Metodologija

Projekt izrade interaktivnog tutorijala započinje idejom te zadavanjem cilja projekta. U ovom slučaju, idejom stvaranja tutorijala gdje će se prikazati određenja znanja iz područja fotografije. Potom slijede faze analize, dizajna i razvoja, implementacije te evaluacija projekta (*ADDIE model*). U okviru ovih faza, određivanje strukture tutorijala bio je korak koji je bio prvi dio organizacije samog tutorijala. U tom koraku, razrađen je poredak blokova, njihova povezanost te njihova svrha. Projekt je dobio dvije podjele unutar kategorija fotografija, a to su boje i kompozicija, te također izbornik koji je imao tri glavne podjele, a to su predavanje (gdje je sažeto prikazano znanje za područja unutar fotografije), primjena (koja se je odnosila na korištenje stečenog znanja kroz provjere unutar kvizova), te galerija kojoj je bila svrha prikazati fotografije i njihove karakteristike. Dijagram niže (Slika 39) prikazuje kako je stvorena

osnovna struktura po kojoj će se graditi tutorijal. Glavni izbornik ima podjelu na tri dijela te korisnik bira na koji želi otići. Sekcije *Predavanja* i *Primjena* imaju dodatno grananje, budući da se sastoje od dva dijela koja sadrže određena znanja.

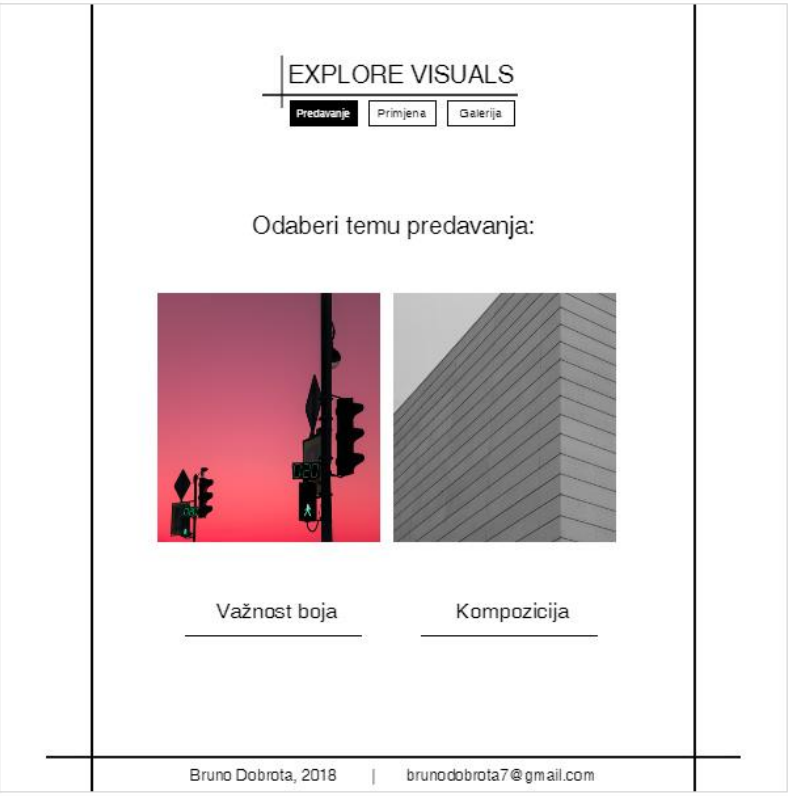


Slika 39: Dijagram funkcioniranja tutorijala

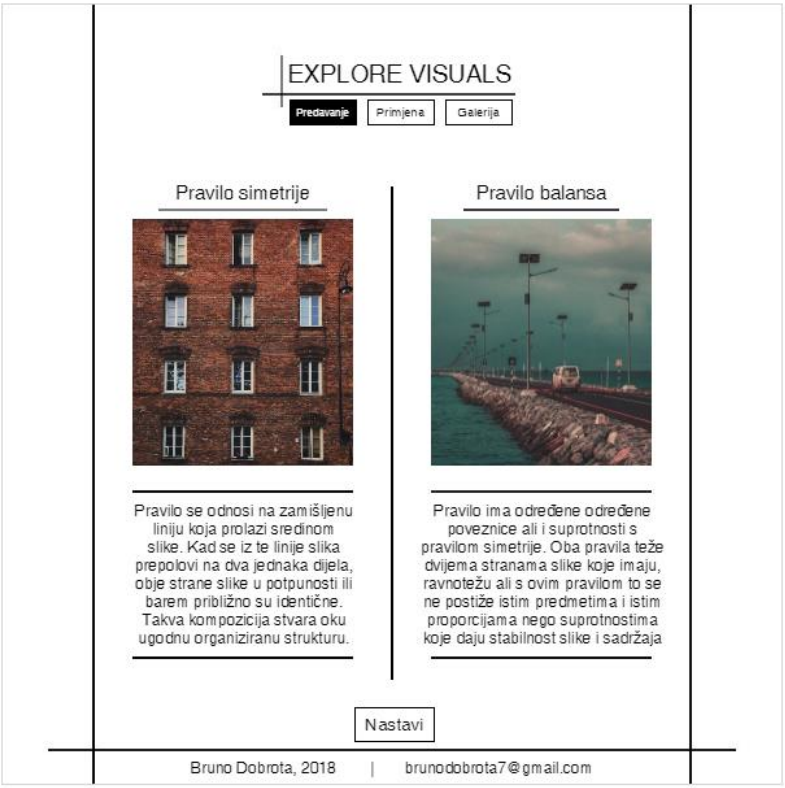
Sljedeće je bilo prikupljanje sadržaja, koji je bio podijeljen na tekstualni te vizualni dio. Tekstualni sadržaj imao je bitnu ulogu za stjecanje znanja, tako da je njegov odabir morao biti pomno proučen i napravljen u količini koja ne bi korisniku predstavljala visoku razinu zahtjevnosti ili predznanja, te da svojom jednostavnošću ne postane zamorno, već suprotno. Vizualni sadržaj odnosi se na slike koje upućuju na tekstualni sadržaj, kao njihov popratni sadržaj koji pokazuje teorijski dio u praksi, odnosno sa svrhom da korisnik lakše shvati određenu informaciju. Također, vizualni sadržaj jest izrazito korišten za kvizove, jer može pomoći pri traženju odgovora ili jest sam jedan od ponuđenih odgovora. Kad se prikupilo dovoljno materijala da se sadržaj može kvalitetno postaviti na stranicu, trebalo je odrediti kakav dizajn će tutorijal imati. Dizajn jest važan dio za multimedijski prikaz znanja, budući da on budi zainteresiranost kod korisnika, te ostavlja određeni dojam na njega. Njegova svrha ima izrazitu važnost. Odlučeno je držati se minimalističkog stila koji će ostaviti dovoljno prostora da vizualni sadržaj poput slika ostane u prvom planu. Nakon što je dizajn određen, trebalo je odabrati alate za implementaciju projekta, te je u tu svrhu korišten *Brackets* uređivač otvorenog pristupa. Kad su svi temelji za kvalitetno stvaranje tutorijala bili pripremljeni, moguće je bilo krenuti sa samom izradom projekta. Budući da je kreiranje iziskivalo veliku količinu sadržaja i vremena, ovaj dio rada se treba smatrati izuzetno važnim. Sama izrada tutorijala svoju kronologiju imala je prateći strukturu projekta, od naslovne te završavajući s kvizom. Nakon što je tutorijal bio gotov, provedena je provjera funkcionalnosti. Po provođenju potrebnih korekcija, tutorijal se može smatrati spremnim za testiranje od strane od potencijalnih korisnika, kako bi mogli uputiti na eventualne dodatne korekcije ili poboljšanja.

Budući da govorimo o tri vrste dizajna (vizualni, interaktivni i instrukcijski), sva tri su obuhvaćena u tutorijalu, budući da se međusobno isprepliću. Vizualni dizajn se odnosi na vizualnu prezentaciju pomoću grafike te primjer (Slika 42) prikazuje korištenje minimalističkog stila u vizualnom dizajnu gdje se dvije paralelne linije sijeku s donjom okomitom crtu, dopuštajući sadržaju koji se nalazi unutar tih crta da ima glavnu riječ, odnosno da ga korisnik prvog primijeti te da se na njemu najviše zadrži. Minimalizam je vrsta pokreta u umjetnosti i dizajnu koja se temelji na ograničenom djelovanju, odnosno manipulacijom nad predmetima te estetskom doživljaju kojeg karakterizira minimalno korištenje boja, linija i sličnih komponenti. Poznat je moto minimalizma „manje je više“ koji ukazuje da činjenica da takav stil ne koristi ili-ili mnogo boja i drugih komponenti, ne znači da ne može biti određeni dizajn vizualno privlačan ili kvalitetan, već naprotiv. Upravo takav stil bio je krucijalan za vizualni identitet tutorijala, da bi sadržaj mogao ostati u prvom planu, a da dizajn privuče korisnika i zainteresira ga. Interaktivni dizajn odnosi se na interakciju korisnika sa sadržajem prikazanim na sučelju. U radu je ova vrsta dizajna jasno vidljiva u načinu na koji je napravljena interakcija u kvizu. Na primjer, povratna informacija na odabrani odgovor ostvarena je tako da se prikazuju križić, odnosno kvačica kao potvrde točnih ili netočnih odgovora na pitanja u kvizu (Slika 32). „Instrukcijski dizajn kao proces uključuje cijeli niz koraka, od analize i determiniranja ciljeva učenja, preko razvoja dizajna i medija koji će se koristiti u edukaciji, pa sve do implementacije i evaluacije materijala koji će se koristiti za potrebe edukacije“ (Jovanović, 2017, str. 4). Primjerice, promišljanje o instrukcijskom dizajnu u početnoj fazi očituje se u samom početku kroz organizaciju sadržaja na dvije glavne teme, za koje se korisnik opredjeljuje.

Na slikama niže vidljiva je organizacija sadržaja na teme i podteme. Dvije glavne teme su važnost boja i kompozicija (Slika 40). Unutar teme „Kompozicija“, sadržane su također dvije teme poput pravila simetrije i pravila balansa (Slika 41). Razlaganje sadržaja tutorijala na manje dijelove, nužno je kako bi korisnik mogao lakše razumjeti sadržaj.



Slika 40: Odabir teme predavanja



Slika 41: Pravila simetrije i balansa

Slika 42: Galerija

6.4 Poboljšanja tutorijala

Budući da izrada ovog i sličnih projekata uključuje različite tipove dizajna, kao što su vizualni dizajn, interaktivni dizajn, te instrukcijski dizajn potrebno je pravovremeno uočiti potencijalnu kompleksnost projekta. S obzirom na to, projekt je moguće višestruko unaprjeđivati. Također, samu početnu ideju bi se moglo bi se razmatrati obzirom na to treba li područje učenja proširiti ili smanjiti, odnosno generalizirati ga ili specificirati. Vizualna komponenta dizajna najviše utječe na to kakav će dojam tutorijal ostaviti na korisnika, koliko će ga privući te zainteresirati da sudjeluje u interaktivnom procesu stjecanja znanja. Također, multimedijski dizajn se može unaprijediti dodatnim elementima kao što su animacija i zvuk. Sveukupno, mjesta za napredak imalo bi određivanje kvalitete i detaljnosti prikazanog sadržaja učenja te same provjere stečenog znanja kako bi korisnik stekao što kvalitetnije znanje.

7. Zaključak

Multimedij se odnosi na mnogo definicija u različitim granama, ali ono što je neizostavno je podjela pojma na multi, odnosno latinski mnogo, te medij, odnosno sredstvo. To je „pojam koji se sastoji od dvije riječi: multi = mnogo; medij = sustav prijenosa i predstavljanja poruke, dakle, predstavljanje korisniku sadržaja koje može istovremeno percipirati koristeći raspoloživa osjetila i mogućnosti (vid, sluh, njuh, opip, okus; pokret, radnja)[2].

Osnovni elementi multimedija mogu se prepoznati kao tekst, zvuk, slika, video, animacija te interakcija. Multimedij kao svoju važnu ulogu ima prikazivanje određenog sadržaja na interaktivan način. Interaktivni sustavi imaju veliku važnost prijenosu znanja.

Multimedijski prikaz znanja može se opisati u kontekstu izrade sustava za učenje koji se temelji na uporabi različitih tipova medija i njihovog kombiniranja radi poboljšanja procesa usvajanja novih znanja i vještina. Obuhvaća područje multimedijskog instrukcijskog dizajna koje uključuje izradu korisničkog sučelja i implementaciju strategija učenja. Tutorijal prikazan u ovom radu temelji se na dva osnovna elementa, a to su tekst i slika. Pri tome je tekst u pisanom, a slika u statičnom obliku. Osnovne faze izrade multimedijskog sustava za učenje odnose se na analizu potreba korisnika, skriptiranje i razvoj multimedijskih materijala i strategija učenja, te implementaciju i evaluaciju sustava u svrhu što boljeg komuniciranja i učenja. Korištenjem tutorijala korisnik može steći znanje u određenom području te je njegovo znanje može biti provjereno pomoću kvizova. Odnosno, kako bi se korisnikova znatiželja i zainteresiranost za učenje povećala, potrebno je dizajnirati sadržaj tutorijala kako bi na interaktivan način motivirao korisnika za njegovo korištenje. Opisani projekt imao je za cilj prikazati znanje iz područja fotografije. Budući da fotografija kao jedno od područja unutar umjetnosti zauzima veliki opseg tema, tutorijal obuhvaća tek osnovne komponente i bitne karakteristike unutar tema boje i kompozicije fotografije. JavaScript omogućuje kvalitetna rješenja vezana za unaprjeđivanje multimedijskog sadržaja koji se plasira na Internet, te u kombinaciji s HTML5 elementom *canvas* omogućuje prikaz različitih sadržaja, poput tutorijala, koji je poslužio kao primjer u ovom završnom radu.

[2] Radić, D., Informatička abeceda. Split, Hrvatska. Preuzeto s: https://informatika.buzdo.com/pojmovi/player-1.htm (21.3.2019.)

Literatura

1. Alexander, E. *Creating a Multiple Choice Quiz in HTML5*. Preuzeto s: http://www.flashbynight.com/tutes/html5quiz/ (25.03.2019.)
2. Chinnathambi, K. (2015). *Working with the Canvas.* Preuzeto s: https://www.kirupa.com/canvas/index.htm (25.03.2019.)
3. Christensson, P. (2014). *JavaScript Definition*. Preuzeto s: https://techterms.com/definition/javascript (25.03.2019.)
4. Crockford, D. (2008). *JavaScript: The Good Part*, Sebastopol, CA.: O'Reilly Media.
5. Flannagan, D. (1997). *JavaScript: The Definitive Guide*. Second Edition. Preuzeto s: https://docstore.mik.ua/orelly/web/jscript/index.html (03.04.2019.)
6. Fulton, J. i Fulton, S. (2011). *HTML5 Canvas: Native Interactivity and Animation for the Web*. Preuzeto s: http://wtf.tw/ref/fulton_fulton.pdf (19.03.2019)
7. Harmadi, V., Kaluža, M., i Liverić, D. (2014). *Usporedba tehnologija prikaza animacija u različitim web-preglednicima*. Zbornik Veleučilišta u Rijeci, 2(1), str. 197-214. Preuzeto s: https://hrcak.srce.hr/128906 (13.04.2019.)
8. Haverbeke, M., (2014). *Eloquent JavaScript*. Preuzeto s: http://eloquentjavascript.net/Eloquent_JavaScript.pdf (03.04.2019.)
9. Hickson, I. (2014). *HTML Living standard*. Preuzeto s: http://www.whatwg.org/specs/web-apps/current-work/multipage/ (30. 3. 2019.)
10. Ivančić, M. (2012). *Interaktivni grafički objekti na Web-u*. Završni rad. Zagreb: Fakultet elektrotehnike i računalstva.
11. Jovanović, S. (2017). *Instrukcijski dizajn kao edukacijski i komunikacijski alat za poboljšanje internih komunikacijskih procesa*. Diplomski rad. Varaždin: Sveučilište Sjever
12. McFarland, D. S. (2014). *JavaScript & jQuery* 3rd ed. Sebastopol, CA.: O'Reilly Media.
13. Musciano, C. i Kennedy, B. (1997). *HTML: The Definitive Guide*. Second Edition. Preuzeto s: https://docstore.mik.ua/orelly/web/html/index.html (13.03.2019.)
14. Pilgrim, M. (2010). *HTML5, Spreman za upotrebu*. Zagreb: Dobar Plan.

15. Powers, S. (2013). *Naučite JavaScript: (obogatite i oživite Web stranice).* Zagreb: Dobar plan.
16. Radić, D., *Informatička abeceda*. Split, Hrvatska. Preuzeto s: https://informatika.buzdo.com/pojmovi/player-1.htm (21.3.2019.)
17. Shannon R. (2012). *What is HTML?*. Preuzeto s: https://www.yourhtmlsource.com/starthere/whatishtml.html (25.03.2019.)
18. Sucan, M. (2009). *HTML5 Canvas – the Basics*. Preuzeto s: https://dev.opera.com/articles/html5-canvas-basics/ (25.03.2019.)
19. *W3schools*. Preuzeto s: https://www.w3schools.com/html/html5_canvas.asp (04.04.2019.)
20. Wagner, G. (2014). *Building Front-End Web Apps with Plain JavaScript*. Preuzeto s: https://oxygen.informatik.tu-cottbus.de/webeng/JsFrontendApp/book/ (09.4.2019.)

Prilozi

JavaScript u multimedijskom prikazu znanja

Sažetak

Programski jezici u tehnološki razvijenom društvu omogućuju razvoj računalnih aplikacija i diseminaciju informacija na različite kreativne načine. Jedan od najviše korištenih programskih jezika koji je usko povezan s domenom interneta, jest JavaScript. U ovom radu biti će prikazana primjena JavaScripta na primjeru izrade multimedijskog tutorijala. Izrada tutorijala uključuje dva tipa dizajna, to su instrukcijski dizajn te dizajn multimedija. Instrukcijski dizajn povezan je s ciljevima i strategijama učenja, dok se dizajn multimedija bavi integracijom različitih oblika medija u multimedijski prikaz znanja. Razvojem programskog koda u JavaScriptu moguće je kvalitetno strukturiranje naredbi i djelotvorno izvršavanje kreativnih ideja zadanih dizajnom projekta. Objašnjavanjem navedenog programskog jezika kroz njegove osnovne karakteristike vezane za glavne funkcije i metode koje sadrži, završni rad će na temelju primjera sadržanih u multimedijskom tutorijalu prikazati načine i svrhu korištenja programskog jezika u implementaciji multimedijskog projekta.

Ključne riječi: JavaScript, HTML, *canvas*, multimedij, multimedijski prikaz znanja, tutorijal

JavaScript in Multimedia Presentation of Knowledge

Summary

Nowadays, programming languages in a hi-tech advanced society allow the development of computer applications and information dissemination in many creative ways. One of the most useful programming languages that is closely connected with the Internet is JavaScript. In this paper, JavaScript usage will be covered by the example of a multimedia tutorial. The tutorial's production includes two types of design – instruction and multimedia design. Instructional design is connected with goals and learning strategies, while multimedia design refers to the integration of various types of media in the multimedia presentation of knowledge. Good quality of structuring statements and efficient executing of creative ideas by the projects' designers are possible with the development of programming code. There will be explained programming language throughout its general characteristics of main functions and methods that tutorial contains. By explaining the programming language through its basic characteristics related to the main functions and methods it contains, this paper will cover ways and purposes of using the programming language in the implementation of a multimedia project, based on the examples contained in the multimedia tutorial.

Key words: JavaScript, HTML, canvas, multimedia, multimedia presentation of knowledge, tutorial