Šolski center Novo mesto
Srednja elektro šola in tehniška gimnazija
Šegova ulica 112
8000 Novo mesto

SPLETNA STRAN ZA RISANJE

(Maturitetna seminarska naloga)

Predmet: računalništvo

Avtor:Adam Tibor Česnik, razred
Mentor: dr. Albert Zorko, univ. dipl. inž. el.
Mentor: Gregor Mede, univ. dipl. inž. rač. in inf.

Novo mesto, april 2022

POVZETEK IN KLJUČNE BESEDE

V seminarski nalogi sem opisal izdelavo spletne strani za risanje, ki sem jo izdelal kot zaključni izdelek za maturo. V prvem delu semiranske naloge na kratko opišem vse uporabljene programske jezike in njihov pomen v seminarski nalogi. V drugem delu naloge pa opišem izdelavo spletne strani. Spletna stran je v računalništvu dokument z nadbesedilom, ki ga prikaže spletni brskalnik. Programiranje spletne strani razdelimo na ospredje in zaledje. V ospredju definiramo njeno strukturo, se pravi elemente, ki bodo uporabniku prikazani. To naredimo s pomočjo programskega jezika HTML. Oblikovanje teh elementov in način prikaza pa opravimo v zaledju, v jeziku CSS. Svoje spletne strani sem se najprej lotil s HTML-jem in jo nato uredil v CSS-ju saj sem glavni program delal v JavaScript-u in sem le tako lahko videl kaj dela in kaj ne. Do moje spletne strani ni mogoče dostopati preko spleta, vendar bom vseeno opisal kako proces kako kupiti svojo domeno ter tako objaviti datoteke na spletnem gostitelju. Šel bom skozi vse potrebne procese za izdelavo in objavo spletne strani na spletu.

Ključne besede: spletna stran, Javascript, HTML, CSS

KAZALA

KAZALO VSEBINE

Contents

KAZALO SLIK

1 UVOD

V tej seminarski nalogi bom izdelal in nato opisal postopek izdelave spletne strani. Namen je ustvariti spletno stran za koga, ki si želi na hitro nekaj preprostega narisati. Temo seminarske naloge sem si izbral ker imam rad izziv, saj še nikoli nisem delal ničesar v JavaSriptu. Ta tema mi je bila tudi mnogo bolj všeč kot kakršnekoli podatkovne baze. Cilj je pridobiti čim več znanja in veščin za ustvarjanje spletne strani s pomočjo označevalnega jezika HTML, oblikovanja ter programiranja drugih funkcionalnosti ki jih omogoča JavaScript, z velikim poudarkom na JavaScript. Za metodo dela sem uporabil predvsem raziskovanje spletne literature.

2 SPLETNA STRAN

2.1 HTML

HTML ali HyperText Markup Language je najbolj osnoven gradnik spleta. Določa pomen in strukturo svetovnega spleta. Druge tehnologije poleg HTML so po navadi uporabljene za spreminjanje izgleda spletnih strani (CSS) in pa njihove funkcionalnosti ali vedenja (JavaScript).

»Hypertext« se navezuje na hiperpovezave ki povezujejo spletne strani druga z drugo, na eni spletni strani ali med različnimi spletnimi stranmi. Hiperpovezave so temeljni vidik spleta. Z nalaganjem vsebine na internet in povezovanjem na strani, ki so jih ustvarili druge ljudje, postanete aktivi udeleženec svetovnega spleta.

HTML uporablja »Markup« za označevanje besedila, slik in druge vsebine za prikaz v spletnem brskalniku. HTML označevanje vključuje posebne elemente, kot so:

<head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, , , <img> in še mnogo drugih.

Element je od drugega besedila v dokumentu ločen z oznakami, ki so sestavljene iz imena elementa, obkroženega z »<« in »>«. Ime elementa znotraj oznake je neobčutljivo na velike in male črke. To pomeni, da je lahko napisano z velikimi, malimi ali mešanico obojih. Na primer, oznako <title> lahko zapišemo tudi kot <TITLE> ali <TiTLe> ali na kakršenkoli drug način. (1)

https://developer.mozilla.org/en-US/docs/Web/HTML

2.2 CSS

Cascading Style Sheets (CSS) je slogovni jezik, ki ga uporabimoo da uredimo izgled dokumenta napisanega v HTML ali XLM. CSS opiše kako naj so elementi upodobljeni na zaslonu.

CSS je med osrednjimi jeziki odprtega spleta in je standardiziran v vseh brskalnikih glede na W3C specifikacije. Prej se je razvoj tazličnih delob specifikcije CSS izvajal sinhrono, kar je omogočalo različico najnovejših priporočil. Poznamo CSS1, CSS2, CSS3, CSS4, vendar CSS4 ni nikoli postal uradna različica.

Od CSS3 se je obseg specifikacije znatno povečal in napredek pri različnih modulih CSS se je začel tako zelo razlikovati, da je postalo učinkovitejše razvijati in izdajati priporočila ločeno za vsak modul. Namesto različice specifikacije CSS, W3C zdaj občasno posname posnetek zadnjega stabilnega stanja specifikacije CSS. (2)

2.3 JAVASCRIPT

Javascript je objektni skriptni programski jezik, ki ga je razvil Netscape, da bi spletnim programerjem pomagal pri ustvarjanju interaktivnih spletnih strani. Jezik je bil razvit neodvisno od Jave, vendar si z njo deli številne lastnosti in strukture. Javascript lahko sodeluje s HTML- kodo in s tem poživi stran z dinamičnim izvajanjem. Javascript podpirajo velika programska podjetja in kot odprt jezik ga lahko uporablja vsakdo, ne da bi pri tem potreboval licenco. Podpirajo ga vsi novejši spletni brskalniki. (3)

Ne zamenjujte JavaScripta s programskim jezikom Java. Tako »Java« kot »JavaScript« sta blagovni znamki ali registrirani blagovni znamki družbe Oracle v ZDA in drugih državah. Vendar imata oba programska jezika zelo različno sintakso, semantiko in uporabo.

JavaScript lahko v datoteko HTML dodate na dva načina:

Notranji JS: JavaScript lahko dodamo neposredno v našo datoteko HTML, tako da napišemo kodo znotraj oznake <script>. Oznako <script> lahko postavite znotraj oznake <head> ali <body> glede na zahtevo.

Zunanji JS: JavaScript kodo lahko zapišemo v drugo datoteko s pripono .js in nato to datoteko povežemo znotraj oznake <head> datoteke HTML, v katero želimo dodati to kodo. (4)

Za svojo seminarsko nalogo sem uporabil zunanji javascript.

2.4 SPLETNO GOSTOVANJE IN OBJAVA SPLETNE STRANI

Storitev spletnega gostovanja je vrsta storitve internetega gostovanja, ki gosti spletna mesta za stranke, to pomeni, da ponuja zmogljivosti, ki so jim potrebne za ustvarjanje in vzdrževanje spletnega mesta, ter omogoča dostop do svetovnega spleta. Podjetja, ki ponujajo storitve spletnega gostovanja, se včasih imenujejo spletni gostitelji.

Do leta 1991 je bil internet omejen na uporabo samo za raziskave in izobraževanje v znanosti in inženirstvu in je bil uporabljen za e-pošto, telnet, FTP in USENET promet, vendar le majhno število spletnih strani. Protokoli svetovnega spleta so bili šele napisani in šele konec leta 1993 ne bi obstajali grafični spletni brskalniki za računalnike Mac ali Windows.

Za gostovanje spletne strani na internetu bi posameznik ali podjetje potrebovalo svoj računalnik ali strežnik.Ker vsa podjetja niso imela proračuna ali strokovnega znanja za to, so storitve spletnega gostovanja začele ponujati gostovanje spletnih mest uporabnikov na njihovih lastnih strežnikih, ne da bi odjemalec moral imeti v lasti potrebno infrastrukturo, potrebno za delovanje spletnega mesta. Lastniki spletnih mest, imenovani tudi spletni skrbniki, bi lahko ustvarili spletno mesto, ki bi gostovalo na strežniku storitve spletnega gostovanja in bi ga storitev spletnega gostovanja objavila v spletu. (5)

2.4.1 KAKO OBJAVITI SVOJO SPLETNO STRAN

Čeprav svoje spletne strani nisem objavil na spletu, ker nisem kupil domene, bom vseeno razložil razložil postopek kako bi lahko spletno stran objavil. Ker to ni najbolj enostavna stvar sem se odločil da postopek opišem s pomočjo 5ih korakov.

Korak 1:

Da uporabnik sploh najde našo spletno stran moramo imeti personalizirano ime oziroma naslov naše spletne strani, kamor gre lahko uporabnik. Strokovno temu rečemo domena spletne strani. Na spletnu najdemo veliko podjetij ki pordajajo svoje domene, npr.:

- GoDaddy.com
- Domains.google
- Namecheap.com

Korak 2:

Najprej si moramo izbrati dobro in zanesljivo podjetje za spletno gostovanje. Ključnega pomena je da izberemo pravega spletnega gostitelja, ki ima vse ključne funkcije za delovanje našega projekta. Tu je nekaj dejavnikov oz. funkcij ki jih moramo upoštevati pri izbiri ponudnika za gostovanje:

- podpora v živo
- nadzor nad našim prostorom za spletno gostovanje
- prostor za rast
- garancija vračila denarja

Korak 3:

Naslednji korak je izbira pravega orodja za nalaganje vaše spletne strani na internet. Obstajajo mnogi načini z objavo različnih pripomočkov kateri ti tudi avtomatsko posodablja spremenjene datoteke. Najbolj popularen je cPanel za objavo datotek na spletno stran. Tega sem tudi jaz uporabil za svojo spletno stran

Korak 4:

Če naše spletno mesto uporablja bazo podatkov (moja spletna stran je ne), jo moramo uvoziti skupaj z datotekami spletnega mesta. To bi naredili tako:

- Ustvarite novo bazo podatkov MySQL in uporabnika
- Dostopajte do vaše baze podatkov prek phpMyAdmin
- Posodobite podrobnosti povezave z bazo podatkov MySQL

S temi štirimi koraki bi morala biti naša spletna stran vzpostavljena in pripravljena na uporabnike ki jo želijo uporabiti. Upam, da je bilo to prikazano na dovolj enostaven in razumljiv način.

3 REALIZACIJA

3.1 STRUKTURA SPLETNE STRANI

Najprej sem definiral strukturo spletne strani v HTML-ju. Kot je že bilo omenjeno, HTML določa vsebino ki je prikazana uporabniku (besedilo, slike, tabele, sezname, gumbe, obrazce itd.). Saj brez HTML datoteke nebi mogel videti ali moj program sploh deluje.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width = device-widt, initial-scale = 1">
        <title> JavaScript Risanje</title>
        <link rel="stylesheet" type="text/css" href="mainstyle.css">
        <script src="risanje_seminarska.js"></script>
    </head>
```

Slika 1 glava HTML dokumenta in povezovanje z CSS in JavaScript

V glavi HTML dokumenta sem povezal HTML datoteko z JavaScript in CSS datoteko, saj če tega nebi naredil, ne bi mogel spletne strani bolj čedno urediti, in program ki sem bi ga napisal v JavaScriptu ne bi mogel nikjer testirati niti videti kako deluje.

```
<link rel="stylesheet" type="text/css" href="mainstyle.css"
```

Z zgornjim ukazom sem HTML document povezal s CSS dokumentom.

Ukaz, da HTML povežemo z JavaScript dokumentom pa izgleda takole:

```
<script src="risanje_seminarska.js"></script>
```

Ti ukazi morajo nujno biti napisani v glavi HTML dokumenta.

Da bomo lahko risali potrebujemo seveda tudi platno na katero bomo risali, to dobimo tako, da že v HTML document vpeljemo “canvas”. Koda za vplejlavo platna izgleda tako:

```
<canvas id="my-canvas" width="600" height="600"></canvas>
```

Nato sem razmislil katere funkcije želim da moj risarski program omogoča. Odločil sem se, da bi rad imel možnosti:

- Čopič
- Črta od točke do točke
- Pravokotnik
- Krog
- Elipsa
- 6-kotnik

Hotel sem da ko kliknemo ikono na zaslonu, da uporabljamo tisto orodje na katerega ikono smo kliknili. To sem naredil s pomočjo onClick ukaza in sicer:

```
<a href="#" id="save" onclick="SaveImage()">
<img src="save-icon.png"></a>
<a href="#" id="brush" onclick="ChangeTool('brush')">
<img src="brush-icon.png"></a>
```

Se pravi, ko kliknemo na sliko, se izvede v java scriptu funkcija SaveImage() ali ChangeTool, odvisno na katero sliko kliknemo. Popolnoma enako sem to naredil za spreminjanje barv, le da sem namesto slikic naredil gumbe. Gumbe sem tudi oblikoval, da ustrezajo barvi katero želimo izbrati (za zeleno bravo v zeleni piše zelena itd.). HTML koda za gumbe za izbiro barv izgleda tako:

```
<a href="#" id="black" onclick="ChangeColor('black')">
<button style="background-color:rgb(230, 230, 230);color:black;border-radius:
5px;border-color:black">Black</button></a>
```

3.2 OBLIKOVANJE SPLETNE STRANI

Kako bodo elementi oblikovno prikazani uporabniku, sem definiral s pomočjo jezika CSS. Določil sem ozadja elementov, barve, odmike od drugih elementov, velikost in vrsto pisave, obrobe itd.

```
7   .toolbar{
8       width: 100%;
9       background-color: rgb(42, 197, 197);
10      overflow: auto;
11  }
12
13  .toolbar a{
14      float: left;
15      width: 13%;
16      text-align: center;
17      padding: 6px 5px;
18      transition: all 0.5s ease;
19  }
20
21  .toolbar a:hover{
22      background-color: white;
23  }
```

Slika 2 CSS oblikovanje

Na drugi sliki vidimo da imamo 3 razrede, **.toolbar** določa lastnosti vrstice za orodja, širina je nastavljena na 100%, saj imamo pred tem že razred .wrapper, ki določa širino na 900px, to lahko naredimo saj so v HTML dokumentu vsi elementi v razredu “wrapper”.

.toolbar a določa lastnosti elementov, v našem primeru sličic, ki so v orodni vrstici.

.toolbar a: hover določa kako se elementi v orodni vrstici odzovejo ko je na njih miška.

3.3 JAVASCRIPT

V programskem jeziku JavaScript sem naredil vse ostalo, se pravi, da lahko sploh rišemo je pomemben JavaScript. Seveda, za imamo kakršenkoli risarski program potrebujemo canvas. Canvas smo že vpeljali v HTML datoteko, vendar ga moramo deklarirati tudi v JavaScriptu.

V prvem delu programa sem uvedel nekaj spremenljivk ki jih bom v programu potreboval. Npr.:

`let canvas;` <- je referenca za canvas

`let ctx;` <- Kontekst ponuja funkcije, ki se uporabljajo za risanje in delo s Canvasom

```
// Shrani predhodno narisane slikovne podatke za obnovitev
// dodane so nove risbe
let savedImageData;
// Shrani, ali trenutno vlečem miško
let dragging = false;
let strokeColor = 'black';
let fillColor = 'black';
let line_Width = 2;
let polygonSides = 6;
// Orodje, ki se trenutno uporablja
let currentTool = 'brush';
let canvasWidth = 600;
let canvasHeight = 600;
```

Slika 3 deklariranje nekaj začetnih spremenljivk in konstant

Vpeljati sem moral tudi nekaj Array-ev. V njih sem shranjeval X in Y koordinate miške, tako da smo lahko sploh risali. V JavaScriptu je Array (matrika) ena sama spremenljivka, ki se uporablja za shranjevanje različnih elementov. Pogosto se uporablja, ko želimo shraniti seznam elementov in do njih dostopati z eno samo spremenljivko.

```
// Shrani, ali trenutno uporabljam čopič
let usingBrush = false;
// Shrani vrstice x & ys, ki se uporabljajo za izdelavo črt čopiča
let brushXPoints = new Array();
let brushYPoints = new Array();
// Shrani, ali je miška pritisnjena
let brushDownPos = new Array();
// Shrani barvo vsake točke
let brushColorPoints = new Array();
```

Slika 4 vpeljevanje Arrayev (matric)

Na sliki 4 vidimo vpeljane matrice in spremenljivko usingBrush ki v nadaljevanju programa preverja če uporabljamo čopič.

Nato sem naredil nekaj razredov, ki vase shranjujejo vrednosti x in y koordinate. To je vidno na slikah 5 in 6.

```
class MouseDownPos{
    constructor(x,y) {
        this.x = x,
        this.y = y;
    }
}
```

Slika 5 razred MouseDownPos

```
class Location{
    constructor(x,y) {
        this.x = x,
        this.y = y;
    }
}
```

Slika 6 razred Location

Na slikah 5 in 6 vidimo 2 razreda ki sta s prvega vidika enaka, vendar ju v nadaljevanju uporabim za 2 različni funkciji, in sicer:

- Razred na sliki 5 drži položaj x in y, kjer je miška kliknjena
- Razred na sliki 6 pa zadržuje x & y lokacijo miške (vzema vrednosti ko miško premikamo)

Da bodo si ti razredi uporabni, sem njihove vrednosti spremenil v spremenljivke tako:

```
let shapeBoundingBox = new ShapeBoundingBox(0,0,0,0);
// Drži položaj x in y, kjer je kliknjen
let mousedown = new MouseDownPos(0,0);
// Zadržuje x & y lokacijo miške
let loc = new Location(0,0);
```

Slika 7 vrednosti razredov spremenil v spremenljivke

Ker potrebujemo da se canvas “naloži” z eventListenerjem kličemo našo funkcijo “setupCanvas” da se izvede, ko se stran naloži.

```
document.addEventListener('DOMContentLoaded', setupCanvas);
```

```
function setupCanvas(){
    // Pridobimo sklic/referenco na element platna
    canvas = document.getElementById('my-canvas');
    // Pridobite metode za upravljanje s platnom
    ctx = canvas.getContext('2d');
    ctx.strokeStyle = strokeColor;
    // Izvede ReactToMouseDown ko je miška pritisnjena
    canvas.addEventListener("mousedown", ReactToMouseDown);
    // Izvede ReactToMouseMove ko je miška pritisnjena
    canvas.addEventListener("mousemove", ReactToMouseMove);
    // Izvede ReactToMouseUp ko je miška pritisnjena
    canvas.addEventListener("mouseup", ReactToMouseUp);
}
```

Slika 8 funkcija setupCanvas

Na sliki 8 vidimo funkcijo setup canvas, ki ustvari platno, na katerega bomo risali. Dodali smo se 3 eventListenerje, ki bodo klicali metode “ReactToMouseDown”, “Move” in pa “Up”, ko miško pritisnemo, jo premaknemo ali pa spustimo.

Da sem spreminjal uporabljena orodja, se pravi ko kliknemo na sliko orodja ki ga hočemo uporabiti na naši spletni strain, moramo povezati HTML z JavaScriptom, to lahko naredimo na nekaj različnih načinov.

METODA	OPIS
document.getElementsById(id)	Poišče element po ID-ju elementa
document.getElementsByTagName(name)	Poišče element po imenu oznake
Document.getElementsByClassName(name)	Poišče element po imenu razreda

Tabela 1: Metode povezovanja HTML-ja in JavaScript

```
function ChangeTool(toolClicked){
    document.getElementById("open").className = "";
    document.getElementById("save").className = "";
```

Slika 9 ChangeTool funkcija

Na sliki 9 vidimo funkcijo “ChangeTool” v Kateri smo deklarirali novo spremenljivko “toolClicked”. S pomočjo metod document.getElementById() izbiramo katero orodje bomo uporabljali, s spremenljivko “toolClicked” pa nastavimo vrednost spremenljivke (naše izbrano orodje) “currentTool” katera je deklarirana izven te funkcije in katere privzeta vrednost je “brush”.

Popolnoma enako sem to naredil za barve, le da imam za barve privzete gumbe in ne slike v orodni vrstici.

Nato smo potrebovali položaj miške glede na položaj platna na naši spletni strani. Do tega sem prišel s pomočjo metode “getBoundingClientRect()”. Metoda Element.getBoundingClientRect() vrne objekt DOMRect, ki zagotavlja informacije o velikosti elementa in njegovem položaju glede na vidno polje. (6) In sicer:

```
function GetMousePosition(x,y){
    //Pridobimo velikost in položaj platna na spletni strani
    let canvasSizeData = canvas.getBoundingClientRect();
    return { x: (x - canvasSizeData.left) * (canvas.width  / canvasSizeData.width),
        y: (y - canvasSizeData.top)  * (canvas.height / canvasSizeData.height)
      };
}
```

Slika 10 položaj miške

Za tem sledi funkcija ki posodobi velikost naše 'škatle' glede na pozicijo miške na platnu.

```
function UpdateRubberbandSizeData(loc){
    shapeBoundingBox.width = Math.abs(loc.x - mousedown.x);
    shapeBoundingBox.height = Math.abs(loc.y - mousedown.y);
    if(loc.x > mousedown.x){
        shapeBoundingBox.left = mousedown.x;
    } else {
        shapeBoundingBox.left = loc.x;
    }
    if(loc.y > mousedown.y){
        shapeBoundingBox.top = mousedown.y;
    } else {
        shapeBoundingBox.top = loc.y;
    }
}
```

Slika 11 shranjevanje lokacij za pomoč pri risanju pravokotnikov

V funkcijo pošljemo spremenljivko “loc” ki nosi x in y koordinate miške. Ta funkcija deluje tako: (mousedown je koordinata, kjer smo miško prvotno kliknili)

- Prvi 2 vrstici: Višina in širina sta razlika med prvotnim klikom in trenutnim položajem miške.
- Prvi if stavek: Če je miška bolj desno, kjer smo prvotno kliknili, shranimo lokacijo mousedown, ker je najbolj levo, če ne shrani lokacijo miške, ker je najbolj levo.

- Drugi if stavek: če je lokacije miške pod mestom, kjer smo prvotno kliknili, shranimo lokacijo mousedown ker je bliže vrhu platna, če ne shranimo položaj miške

6-kotnika sem se lotil tako, da sem sestavil več trikotnikov skupaj, da mi je to uspelo sem si pomagal s trigonometrijo.

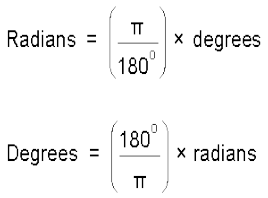
```
function getAngleUsingXAndY(mouselocX, mouselocY){
    let adjacent = mousedown.x - mouselocX;
    let opposite = mousedown.y - mouselocY;

    return radiansToDegrees(Math.atan2(opposite, adjacent));
}
```

Slika 12 dobljeni kot

Na sliki 12 vidimo funkcijo, ki nam vrne kot tako: Vrne kot z uporabo x in y, x = priležna stranica, y = nasprotna stranica, Tan(kot) = nasprotna / priležna, Kot = ArcTan (nasprotna / priležna)

Vrne nam vrednost v stopinjah, s pomočjo naslednje funkcije:

```
function radiansToDegrees(rad){
    if(rad < 0){
        return (360.0 + (rad * (180 / Math.PI))).toFixed(2);
    } else {
        return (rad * (180 / Math.PI)).toFixed(2);
    }
}
```



Slika 13 formula za radiane v stopinje in obratno

Slika 14 radiane v stopinje

Na sliki 13 vidimo spreminanje radianov v stopinje po naslednji formuli (glej sliko 14), da se izognemo negativnim kotom, dodamo if stavek, v katerem, če je kot negativen mu prišteje 360 stopinj.

```
function degreesToRadians(degrees){
    return degrees * (Math.PI / 180);
}
```

Slika 15 stopinje v radiane

Da dobimo točke za 6-kotnik moramo tudi iz stopinj pretvoriti v radiane, zakaj pa vam pokažem zdaj.

Funkcijo getPolygonPoints bom razložil v 2eh delih.

```
function getPolygonPoints(){
    let angle =  degreesToRadians(getAngleUsingXAndY(loc.x, loc.y));
    let radiusX = shapeBoundingBox.width;
    let radiusY = shapeBoundingBox.height;
    let polygonPoints = [];
```

Slika 16 1. del funkcije get polygon points

Na sliki 16 s prvo vrstico pridobimo kot v radianih na podlagi x in y lokacije miške, druge 2 vrstici shranita X & Y za točko X & Y, ki predstavlja polmer,ki je enak X in Y mejne škatle, zadnja vrstica pa shrani vse točke v 6-kotniku.

```
for(let i = 0; i < polygonSides; i++){
    polygonPoints.push(new PolygonPoint(loc.x + radiusX * Math.sin(angle),
    loc.y - radiusY * Math.cos(angle)));
    angle += 2 * Math.PI / polygonSides;
}
return polygonPoints;
```

Slika 17 2. del funkcije get polygon points

Vsako točko v poligonu najdemo tako, da prelomimo dele mnogokotnika v trikotnike Potem lahko uporabimo znani kot in dolžino sosednje strani da najdemo X = mouseLoc.x + radiusX * Sin(kot)

Najdemo Y = mouseLoc.y + radiusY * Cos(angle)

Ostane nam le še to, da narišemo naš 6-kotnik.

```
function getPolygon(){
    let polygonPoints = getPolygonPoints();
    ctx.beginPath();
    ctx.moveTo(polygonPoints[0].x, polygonPoints[0].y);
    for(let i = 1; i < polygonSides; i++){
        ctx.lineTo(polygonPoints[i].x, polygonPoints[i].y);
    }
    ctx.closePath();
}
```

Slika 18 narišemo 6-kotnik

Slika 18 kaže kodo, s katero sem povezal vseh 6 točk katere smo dobili s prejšnjo funkcijo.

Sledi –dolg if stavek ki preveri katero orodje imamo izbrano.

```
function drawRubberbandShape(loc){
    ctx.strokeStyle=strokeColor;
    if(currentTool === "brush"){
        // Ustvari čopič
        DrawBrush();
    } else if(currentTool === "line"){
        // Nariše črto
        ctx.beginPath();
        ctx.moveTo(mousedown.x, mousedown.y);
        ctx.lineTo(loc.x, loc.y);
        ctx.stroke();
    } else if(currentTool === "rectangle"){
        // Ustvari pravokotnike
        ctx.strokeRect(shapeBoundingBox.left, shapeBoundingBox.top, shapeBoundingBox.width, shapeBoundingBox.height);
    } else if(currentTool === "circle"){
        // Ustvari kroge
        let radius = shapeBoundingBox.width;
        ctx.beginPath();
        ctx.arc(mousedown.x, mousedown.y, radius, 0, Math.PI * 2);
        ctx.stroke();
    } else if(currentTool === "ellipse"){
        // Ustvarite elipse
        // ctx.ellipse(x, y, radiusX, radiusY, rotacija, startAngle, endAngle)
        let radiusX = shapeBoundingBox.width / 2;
        let radiusY = shapeBoundingBox.height / 2;
        ctx.beginPath();
        ctx.ellipse(mousedown.x, mousedown.y, radiusX, radiusY, Math.PI / 4, 0, Math.PI * 2);
        ctx.stroke();
    } else if(currentTool === "polygon"){
        // Ustvari mnogokotnike
        getPolygon();
        ctx.stroke();
    }
}
```

Slika 19 izbrano orodje

Na sliki 19 vidimo if stavek ki preverja katero orodje imamo izbrano. V funkcijo pošljemo spremenljivko »loc«, tako da zaznamo lokacijo miške, in kje je bila miška kliknjena. Funkcijo »drawBrush« bomo naknadno dodali, vsa druga orodja pa bi morala sedaj delovati.

```
function UpdateRubberbandOnMove(loc){
    UpdateRubberbandSizeData(loc);
    drawRubberbandShape(loc);
}
```

Slika 20 premikanje med risanjem

Ta funkcija nam omogoča da vidimo pravokotnike, 6-kotnike, elipse, kroge in daljice še preden spustimo miško. To pomeni da vidimo vmes kako bo program narisal te oblike, ne pa miško kliknemo ter povlečemo in medtem ne vidimo nič, šele ko miško spustimo pa da se nam pokaže lik ki smo ga narisali.

```
function AddBrushPoint(x, y, mouseDown){
    brushXPoints.push(x);
    brushYPoints.push(y);
    // Shrani, da je miška pritisnjena
    brushDownPos.push(mouseDown);

    brushColorPoints.push(strokeColor);
}
```

Slika 21 dodamo x in y točke

S funkcijo na sliki 21 v matrice shranimo trenutne koordinate x,y in ali je miška pritisnjena. Shranimo tudi barvo točke.

Funkcijo »DrawBrush()« sem naredil s pomočjo for zanke:

```
function DrawBrush(){
    for(let i = 1; i < brushXPoints.length; i++){
        ctx.lineCap = 'round';
        ctx.beginPath();

        ctx.strokeStyle = brushColorPoints[i];

        // Preverite, ali je bil gumb miške na tej točki pritisnjen
        // in če je tako, nadaljujte z risanjem
        if(brushDownPos[i]){
            ctx.moveTo(brushXPoints[i-1], brushYPoints[i-1]);
        } else {
            ctx.moveTo(brushXPoints[i]-1, brushYPoints[i]);
        }
        ctx.lineTo(brushXPoints[i], brushYPoints[i]);
        ctx.closePath();
        ctx.stroke();
    }
}
```

Slika 22 čopič

Na sliki 22 gremo s funkcijo »DrawBrush()« čez vse točke čopiča in jih povežemo. Povežemo pa jih z ukazom `ctx.lineTo(brushXPoints[i], brushYPoints[i]);`

```
function ReactToMouseDown(e){
    canvas.style.cursor = "point";
    loc = GetMousePosition(e.clientX, e.clientY);
    SaveCanvasImage();
    mousedown.x = loc.x;
    mousedown.y = loc.y;
    dragging = true;
    if(currentTool === 'brush'){
        usingBrush = true;
        AddBrushPoint(loc.x, loc.y);
    }
}
```

Slika 23 react to mouse down

V odseku kode na sliki 23 najprej spremenimo kazalec miške v križec, nato shranimo lokacijo kje smo miško pritisnili in shranimo trenutno sliko platna. Nato shranimo položaj miške ob kliku in shranimo da miško držimo pritisnjeno. V if stavku pa bo čopič shranil točke v niz.

```
function ReactToMouseMove(e){
    canvas.style.cursor = "crosshair";
    loc = GetMousePosition(e.clientX, e.clientY);
    if(currentTool === 'brush' && dragging && usingBrush){
        if(loc.x > 0 && loc.x < canvasWidth && loc.y > 0 && loc.y < canvasHeight){
            AddBrushPoint(loc.x, loc.y, true);
        }
        RedrawCanvasImage();
        DrawBrush();
    } else {
        if(dragging){
            RedrawCanvasImage();
            UpdateRubberbandOnMove(loc);
        }
    }
}
```

Slika 24 react to mouse move

Na sliki 24 imamo funkcijo ki nam pove kaj storiti ko se miška premika. If stavek pravi da, če uporabljamo za orodje čopič in vlečemo, potem shrani vsako točko, če ne, kliče funkcijo ki nam omogoča da vidimo kaj rišemo, preden shrani te točke (katerokoli drugo orodje razen čopiča) if stavek v prvotnem if stavku pa odvrže slike s čopičem ki se pojavijo zunaj platna.

```
function ReactToMouseUp(e){
    canvas.style.cursor = "default";
    loc = GetMousePosition(e.clientX, e.clientY);
    RedrawCanvasImage();
    UpdateRubberbandOnMove(loc);
    dragging = false;
    usingBrush = false;
}
```

Slika 25 react to mouse up

Funkcija na sliki 25 se izvede ko spustimo miško. Se pravi dobi lokacijo kje smo spustili miško, ponovno nariše canvas in nariše obliko ki smo jo izbrali (če uporabljamo vsa orodja razen čopiča, čopič riše sproti).

```
function SaveImage(){
    var imageFile = document.getElementById("img-file");
    imageFile.setAttribute('download', 'image.png');
    imageFile.setAttribute('href', canvas.toDataURL());
}
```

Slika 26 shranimo sliko

S to funkcijo shranimo sliko v naš privzeti imenik za prenose. S prvo vrstico pridobimo sklic na element povezave, z drugo vrstico nastavimo, da želimo prenesti sliko, ko kliknemo na povezavo. Shranimo pa karkoli je trenuno na platnu.

Sledila bi še objava spletne strani vendar jaz tega nisem naredil ker je to kar udarec za moj trenutni proračun.

4 ZAKLJUČEK

Spletna stran je sestavljena iz ospredja, ki definira strukturo spletne strani in oblikuje elemente, ki so uporabniku prikazani ter iz zaledja, ki uporabniku ni viden, vendar je zelo pomemben. Brez zaledja bi bila moja spletna stran popolnoma neuporabna. Največ težav mi je povzročal javaScript saj ga v šoli nismo uporabljali. Čeprav je zelo podoben sami javi je vseeno ravno toliko drugačen da je prihajalo do veliko sintaksnih napak in sem se s tem kar lovil. S pomočjo HTML sem z znanjem ki sem ga pridobil v šoli vzpostavil spletno stran in jo s programskim jezikom CSS oblikoval. Pri tem nisem imel težav. Kar nekaj težav pa se mi je porajalo pri javaScriptu saj je bilo to prvič da sem delal z canvasom in risanjem. Veliko znanja o javaScriptu sem odnesel z raznih forumov in youtube posnetkov. Tako sem vzpostavil delujočo spletno stran na kateri lahko rišemo in s tem je cilj seminarske naloge dosežen. Nalogo bi naprej lahko razvijal tako, da bi dodal še možnost spreminjanja debeline čopiča in tako da bi kupil domeno ter spletno stran objavil.

5 ZAHVALA

Rad bi se zahvalil profesorjema in mentorjema Gregorju Medetu in Albertu Zorku, ki sta me učila vsa 4 leta srednje šole. Pri njiju sem se naučil da se brez muje še čevelj ne obuje. Tu sem se naučil programiranja in se s pomočjo njiju tudi odločil o svojem nadalnjem šolanju.

Rad bi se zahvalil tudi prijatelju Niku, ki mi je vedno priskočil na pomoč ko se mi je pri čem zataknilo.

Zahvaljujem se tudi svoji družini ki mi je vsa 4 leta stala ob strani in me spodbujala da dosežem svoj potencial.

6 VIRI IN LITERATURA

1. mozzila. *https://developer.mozilla.org/en-US/docs/Web/HTML .* [Online]

2. contributors, MDN. mozilla css. *https://developer.mozilla.org/en-US/docs/Web/CSS.* [Online] mar 2022.

3. Grobelnik, Damnja. damjan js. *http://www.damijan.si/javascript/.* [Online] 2022.

4. geeks for geeks. *https://www.geeksforgeeks.org/javascript/ .* [Online]

5. Spletna Stran Wikipedia. *https://en.wikipedia.org/wiki/Web_hosting_service#:~:text=A%20web%20hosting%20service%20is,are%20sometimes%20called%20web%20hosts.* [Online] oct 2022.

6. contributors, MDN. getBoundingClientRect. *https://developer.mozilla.org/en-US/docs/Web/API/Element/getBoundingClientRect.* [Online] feb 2022.

7. —. Mozilla . *https://developer.mozilla.org/en-US/docs/Web/HTML.* [Online] feb 2022.

Zelo so mi tudi pomagali forumi kot so:

- Geeks for geeks -> https://www.geeksforgeeks.org
- W3schools -> https://www.w3schools.com
- Stack Overflow -> https://stackoverflow.com

7 STVARNO KAZALO

8 PRILOGE

Priloga 1- povezava do programske kode, word dokumenta, pdf dokumenta in power point dokumenta se nahaja na spletnem mestu:

https://github.com/aTc03/seminarska-naloga