Interaktivni grafički objekti u Canvasu i Kinetic.js

Margarić, Natalija

Sveučilište Jurja Dobrile u Puli

Fakultet informatike

Natalija Margarić

Interaktivni grafički objekti u HTML 5 i KineticJS

Završni rad

Pula, 2019.

Sveučilište Jurja Dobrile u Puli

Fakultet informatike

Interaktivni grafički objekti u HTML 5 i KineticJS

Završni rad

Natalija Margarić

Redovan student

Studijski smjer: informatika

Kolegij: Dinamičke web aplikacije

Mentor: Prof. dr. sc. Mario Radovan

Komentor: dr.sc. Nikola Tanković

Pula, rujan 2019

IZJAVA O AKADEMSKOJ ČESTITOSTI

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

Student

_____

U Puli, _____, _____ godine

.

IZJAVA

o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom

_____

_____

__ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

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

U Puli, _____ (datum)

Potpis

_____

Sadržaj

1. Uvod

Internet je u početku bio samo tekst, što je bilo monotono, tako da su slike uvedene prvo putem elementa <img>, a kasnije putem CSS svojstava kao što su pozadinske slike i SVG (Scalable Vector Graphics). Danas, uz brzo širenje različitih mobilnih uređaja, isporuka brzih i robusnih web stranica s podrškom za multimediju važnija je nego ikad prije. Krajnjem korisniku pružiti visoku razinu interaktivnosti i multimedije više nije samo lijepi dodatak, nego je postao standard i norma. Već duže vrijeme imamo dodatke poput Flasha, Silverlight ili Java odgovorni za poslove integracije medija. Bili su dobri prije nekog vremena, kada su stolna računala dominirala na čitavom digitalnom tržištu no, Html5 dodaje nove značajke koje mogu pružiti bogatu interaktivnost bez instaliranja dodatnih dodataka. Jedna od tih novih značajki je element platna (eng. Canvas). Danas kada su preglednici počeli podržavati element <canvas> i povezani Canvas API (Apple ga je izumio oko 2004. godine) i drugi su preglednici počeli pratiti njegovu primjenu u godinama koje su uslijedile. Kao što ćemo vidjeti u nastavku, platno nudi mnogo korisnih alata za stvaranje 2D animacija, igara, vizualizacije podataka i drugih vrsta aplikacija, posebno u kombinaciji s nekim drugim API – jevima koje pruža web platforma.[1]

Osnovni HTML5 Canvas API uključuje 2D kontekst koji programeru omogućuje crtanje različitih oblika, prikaz teksta i prikazivanje slika izravno na određeno područje prozora preglednika. Mogu se primijeniti boje, rotacije, manipulacije piksela i razne vrste linija, krivulja, kutija i ispuna za povećanje oblika, teksta i slika koji se postavljaju na platno, a koje će biti prikazane kroz jednostavne primjere. Prikazat ćemo kako definirati jednostavan element platna i kako izvoditi neke primarne operacije u JavaScript-u što nam pomaže u modificiranju tog elementa i nekoliko primjera koji uključuju pravokutnike, lukove, prilagođene oblike, slike i sofisticirane krivulje.

Biblioteka KineticJS koju ćemo prikazati je neizostavna jer proširuje HTML5 canvas i omogućuje njegovu interaktivnost i tako doprinosi modernom izgledu aplikacija, web stranica i ostalih sadržaja na internetu.

2. Osnove i uvod u HTML 5

HTML 5 je peta verzija HTML-a, jezika za predstavljanje i strukturiranje web dokumenta. To je jezik za organiziranje i prikazivanje sadržaja na WWW-u nastao kao rezultat rada organizacije W3C (World Wide Web Consortium - organizacija koja se bavi standardizacijom tehnologija korištenih na webu), a sa željom da se poboljša i olakša organizacija i prikazivanje sadržaja prisutnih na WWW-u danas i u budućnosti, te da se omogući pristup istima na jednak način s različitih preglednika, odnosno uređaja.[1] HTML5 ne pripada niti jednoj tvrtki niti pojedinom pregledniku, ali ga koriste velike kompanije poput Google-a, Microsoft-a, Apple-a, Mozzile i drugih.[2] Korištenjem HTML5-a možemo web dokument mnogo smislenije čitati. HTML5 je podržan u svim modernim preglednicima. Uz to, svi preglednici, stari i novi, automatski ručno identificiraju nepriznate elemente kao umetnute elemente. Zbog toga starije preglednike možemo "naučiti" da rukuju s "nepoznatim" HTML elementima. S nekoliko novih oznaka možemo s lakoćom poboljšati korisničko iskustvo web dokumenta. Možemo koristiti audio, video zapise, oznake izvora za lako predstavljanje multimedijskog sadržaja. Korištenjem platnene oznake (eng. canvasa) možemo napraviti svoj web dokument crtež crteža.[3]

HTML5 nudi nekoliko novih oznaka i atributa za poboljšanje korisničkog iskustva u obrascima. Možemo razviti izvrsne web aplikacije s HTML5 i srodnim API-jevima poput:[4]

- HTML Geolocation
- HTML Drag and Drop
- HTML Local Storage (lokalna pohrana)
- HTML Application Cache
- HTML Web Workers
- HTML SSE (Server-Sent Events)

S HTML5 i JavaScript knjižnicama čak možemo razviti mobilne aplikacije u obliku web aplikacija i zatim ih prebaciti na matičnu platformu (na primjer iPhone aplikacije). Dakle, s HTML5 možemo stvoriti izvrsne web aplikacije, mobilne aplikacije, pa čak i igre

temeljene na pregledniku. Dizajnirana je tako da je mogu koristiti svi programeri otvorenog weba. Spomenuti ćemo nekoliko značajki razvrstanih u grupe:

- Semantika: omogućava preciznije opisivanje svog sadržaja.
- Povezivanje: omogućava komunikaciju s poslužiteljem na nove i inovativne načine.
- Izvan mrežno i pohranjivanje: omogućava web stranicama lokalno pohranjivanje podataka na strani klijenta i učinkovitije funkcioniranje izvan mrežno.
- Multimedija: izrada video i audio građana prvog razreda na otvorenom webu.
- 2D / 3D grafika i efekti: omogućuju mnogo raznolikiji izbor mogućnosti prezentacije.
- Učinkovitost i integracija: pružanje veće optimizacije brzine i bolje korištenje računalnog hardvera.
- Pristup uređaju: omogućava upotrebu različitih uređaja za ulaz i izlaz.
- Dizajn: dopuštanje autorima pisanje sofisticiranijih tema.[5]

Prednosti:

- Svi preglednici podržani.
- Jednostavan za uporabu i implementaciju.
- HTML 5 u integraciji s CSS-om, JavaScript-om može pomoći u stvaranju prekrasnih web stranica.

Nedostaci:

Treba napisati dugačke kodove što zahtijeva mnogo vremena.[6]

3. Općenito o canvasu

Canvas je platno koje se može koristiti za prikazivanje grafikona, grafike igara ili drugih vizualnih slika u pokretu. Jednostavno rečeno, uz pomoć JavaScripta i HTML5 platna možemo stvoriti 2D oblike i bitmap slike. Platno (eng. canvas) je jedna od najtraženijih značajki u HTML5. Programeri ga vole koristiti za izradu bogatih web aplikacija. Korisnici mogu upotrebljavati te aplikacije bez upotrebe vlastitih dodataka preglednika kao što je Adobe-ov flash player. Većina modernih preglednika kao što su

Chrome, Firefox, Safari i Opera podržavaju ga. Platno se temelji na bitmapu. Omogućuje crtanje grafike i oblika putem JavaScripta. Na raspolaganju je nekoliko JavaScript biblioteka koje olakšavaju rad s Canvasom, a to su: [7]

- Kinetic.js
- Easel.js
- Fabric.js
- Zebra

Prednosti platna:

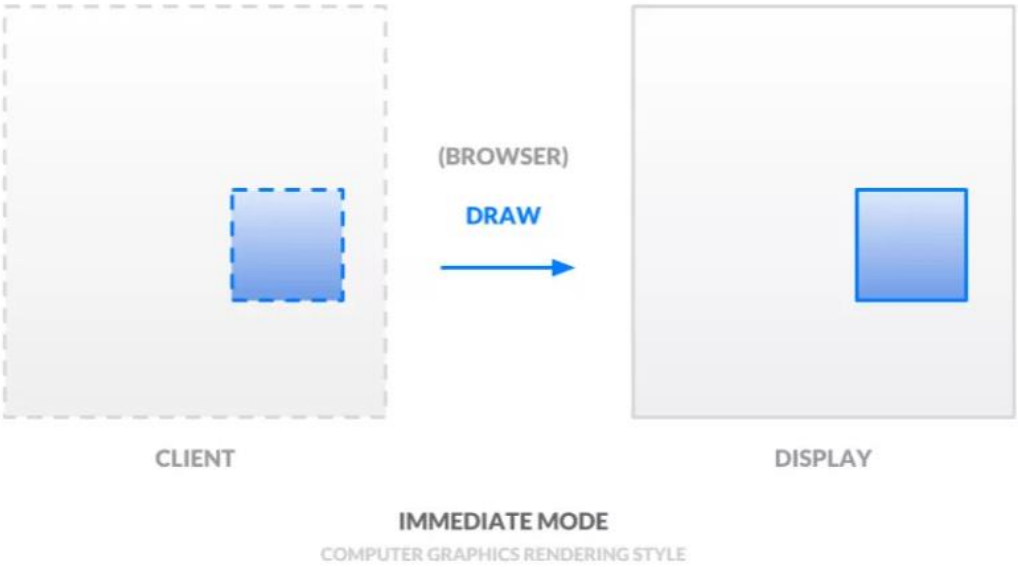
- Brz i fleksibilan
- Sitno zrnata kontrola vremena, kada i kako se prikazuju pikseli
- Hardversko ubrzanje za sva prikazivanja, animacije itd.
- Odličan je za složenu grafiku i kada ima mnogo objekata kojima se može manipulirati.

Nedostatci:

- Manje apstrakcije (ali to se može prevladati pomoću knjižnica trećih strana).
- Nema ugrađenih interakcija na razini objekta, jer platno radi na razini piksela.
- Sporo pri crtanju velikih predmeta ili pri upravljanju velikim platnom.
- Ne podešava se automatski za omjer piksela, što rezultira mutnim tekstom.
- Povećana složenost koda za animacije. [8]

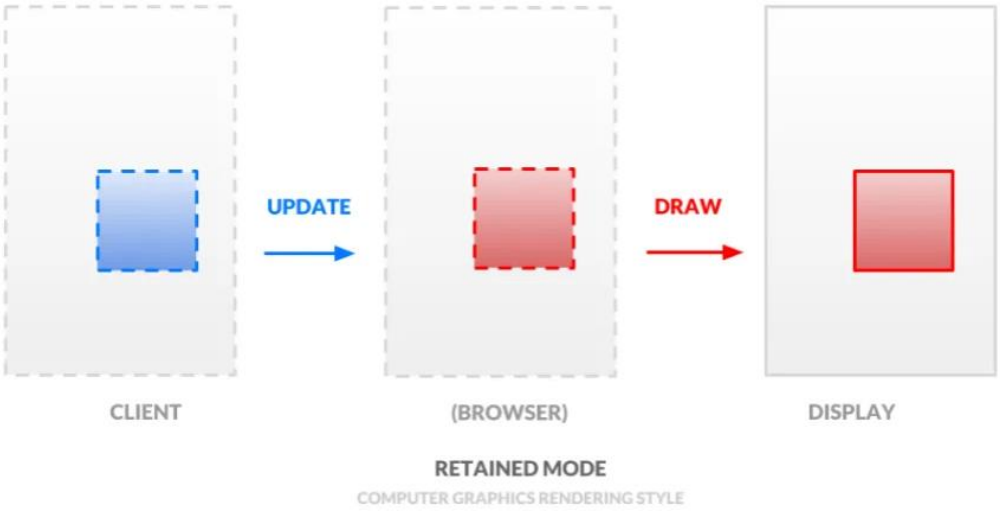
3.1. Računalna grafika

Da bismo objasnili djelotvornost platna, moramo razlikovati dva stila prikazivanja u računalnoj grafici: neposredni način rada i zadržani način rada, koji predstavljaju Canvas i DOM. U neposrednom načinu rada klijent izdaje pozive koji rezultiraju neposrednim prikazom grafičkih objekata. Svaki poziv funkcije platna rezultira grafičkim objektom koji je odmah nacrtan. Bez obzira na to koji se dijelovi platna ažuriraju, cijelo se platno mora svaki put iznova crtati. To znači da klijent mora održavati model predmeta na platnu. U nastavku plava boja predstavlja piksele, a API zove razvojnog programera izravno.



Slika 1. Prikaz neposrednog načina rada u računalnoj grafici

Suprotno tome, u zadržanom načinu rada pozivi klijenta ne rezultiraju trenutnim prikazom grafičkih objekata. Umjesto toga, klijent poziva ažuriranje internog modela. U našem slučaju programeri određuju HTML i pridruženi CSS, a preglednik rukuje kada i kako prikazati ove grafike. To omogućava pregledniku da dodatno optimizira kada i koji su objekti prikazani. U nastavku plava boja (opet) predstavlja API koji vas poziva kao razvojnog programera izravno. Međutim, crvena predstavlja apstrakcije s kojima preglednik rukuje, uključujući održavanje internog modela i ažuriranje grafičkih objekata. [8]



Slika 2. Prikaz zadržanog načina rada u računalnoj grafici

Bivši, neposredni način rada, nudi fleksibilnost, ali zahtijeva miješanje u pojedinosti. A ovaj potonji, zadržani način rada, ograničava fleksibilnost, ali apstrahira daleko detalje. Ove paradigme objašnjavaju filozofije iza platna i DOM-a.

4. Crtanje 2d objekata [6]

Opisati ćemo kako pomoću <canvas> elementa crtati 2D grafiku, počevši od osnova. Navedeni primjeri će nam pokazati što se sve može učiniti s platnom i pružit će isječke koda koji mogu pomoći u daljnjem razvijanju.

U HTML-u, element <canvas> izgleda ovako:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Element <canvas> mora imati atribut id da bi ga JavaScript mogao koristiti. Atribut širine i visine potreban je za definiranje veličine platna.[7]

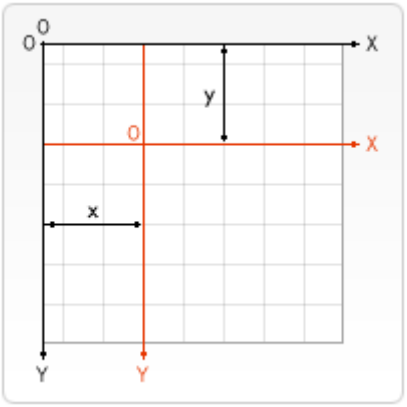
4.1. Transformacije u canvasu

[9] Transformacije se mogu primijeniti na sve crteže na HTML5 platnu. Ovdje je popis transformacija koje se mogu primijeniti:

- translacija (pomicanje onoga što je nacrtano)
- rotacija
- skaliranje

4.1.1 Translacija

Prva od metoda transformacije koju ćemo opisati je translate (). Ova metoda se koristi za premještanje platna i njegovog podrijetla u drugu točku rešetke.



Slika 3. Grafički prikaz translacije

Metoda translate() dodaje transformacijsku transformaciju trenutnoj matrici pomicanjem platna i njegovih izvornih x jedinica vodoravno, a y jedinica okomito na rešetku.

- X - Udaljenost za pomicanje u vodoravnom smjeru. Pozitivne vrijednosti su desno, a negativne lijevo.
- Y - Udaljenost za pomicanje u okomitom smjeru. Pozitivne vrijednosti su smanjene, a negativne gore.[10]

Dobro je sačuvati stanje platna prije bilo kakvih transformacija. U većini slučajeva jednostavno je pozvati metodu vraćanja nego učiniti obrnuti prijevod kako bi se vratili u prvobitno stanje. Također, ako se prevodi unutar petlje i ne spremi se i ne vrati stanje platna, možda će nedostajati jedan dio crteža, jer je nacrtan izvan ruba platna. [11]

Primjer translacije:

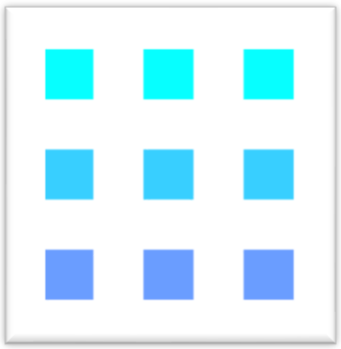
Ovaj primjer pokazuje neke od prednosti translacije porijekla platna. Bez metode translate (), svi bi se pravokutnici crtali na istoj poziciji (0,0). Metoda translate () također nam daje slobodu da pravokutnik postavimo bilo gdje na platnu bez ručnog podešavanja koordinata u funkciji fillRect (). To ga čini malo lakšim za razumijevanje i korištenje. U funkciji draw () pozivamo funkciju fillRect () devet puta koristeći dvije za petlje. U svakoj se petlji prevodi platno, crta se pravokutnik i vraća se u prvotno stanje.

Treba obratiti pažnju na to kako poziv za fillRect () koristi iste koordinate svaki put, oslanjajući se na translate () za podešavanje položaja crtanja. Primjer koda:

```
function draw() {
  var ctx = document.getElementById('canvas').getContext('2d');
  for (var i = 0; i < 3; i++) {
    for (var j = 0; j < 3; j++) {
      ctx.save();
      ctx.fillStyle = 'rgb(' + (51 * i) + ', ' + (255 - 51 * i) + ',
255)';
      ctx.translate(10 + j * 50, 10 + i * 50);
      ctx.fillRect(0, 0, 25, 25);
      ctx.restore();
    }
  }
```

Slika 4. Prikaz koda za translaciju

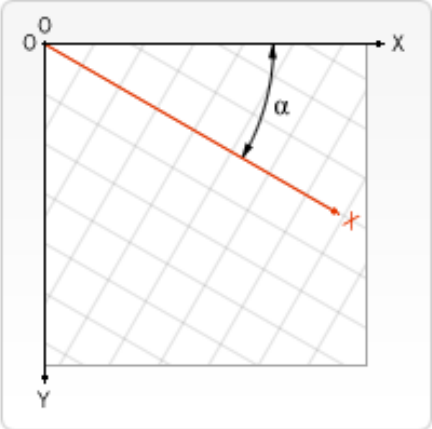
Rezultat koda:



Slika 5. Prikaz izvođenja koda

4.1.2. Rotacije

Druga metoda transformacije je rotate (rotacija). Koristimo ga za okretanje platna oko trenutnog podrijetla. Rotira platno u smjeru kazaljke na satu oko trenutnog podrijetla za broj kutnih radijana.



Slika 6. Grafički prikaz rotacije

Središnja točka rotacije je uvijek u podrijetlu platna. Da bismo promijenili središnju točku, mora se premjestiti platno pomoću metode translate ().

Primjer rotacije:

U ovom ćemo primjeru upotrijebiti metodu rotate () kako bi prvo zaokrenuli pravokutnik iz izvora platna, a zatim iz središta samog pravokutnika uz pomoć translate ().

Primjer koda:

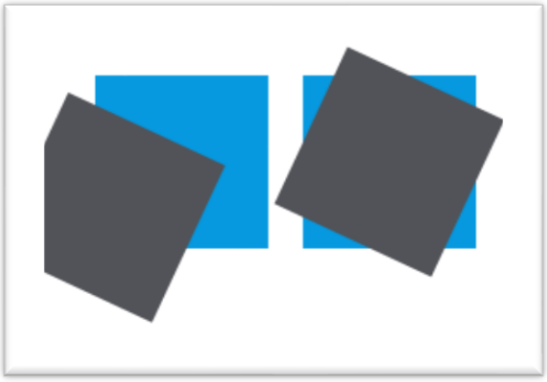
```
function draw() {
  var ctx = document.getElementById('canvas').getContext('2d');

  // left rectangles, rotate from canvas origin
  ctx.save();
  // blue rect
  ctx.fillStyle = '#0095DD';
  ctx.fillRect(30, 30, 100, 100);
  ctx.rotate((Math.PI / 180) * 25);
  // grey rect
  ctx.fillStyle = '#4D4E53';
  ctx.fillRect(30, 30, 100, 100);
  ctx.restore();
```

```
  // right rectangles, rotate from rectangle center
  // draw blue rect
  ctx.fillStyle = '#0095DD';
  ctx.fillRect(150, 30, 100, 100);

  ctx.translate(200, 80); // translate to rectangle center
                          // x = x + 0.5 * width
                          // y = y + 0.5 * height
  ctx.rotate((Math.PI / 180) * 25); // rotate
  ctx.translate(-200, -80); // translate back

  // draw grey rect
  ctx.fillStyle = '#4D4E53';
  ctx.fillRect(150, 30, 100, 100);
}
```

Slika 7. Prikaz koda za rotaciju

Kako bi rotirali pravokutnik oko njegovog središta, prelazimo preko platna u središte pravokutnika, zatim rotiramo platno i prelazimo preko platna natrag u 0,0, a zatim crtamo pravokutnik.

Rezultat koda:



Slika 8. Prikaz izvođenja koda za rotaciju

4.1.3 Skaliranje

Sljedeća metoda transformacije je skaliranje. Koristimo ga za povećanje ili smanjenje jedinica u našoj platnenoj mreži. To se može koristiti za crtanje smanjenih ili povećanih oblika i bitmapova. Mijenja veličine platna po x vodoravno i y okomito. Oba parametra su stvarni brojevi. Vrijednosti manje od 1,0 smanjuju veličinu jedinice,

a vrijednosti iznad 1,0 povećavaju veličinu jedinice. Vrijednosti 1,0 ostavljaju jedinice iste veličine. Pomoću negativnih brojeva može se izvršiti zrcaljenje osi. Prema zadanim postavkama, jedna jedinica na platnu iznosi točno jedan piksel. Primijenimo li na primjer faktor skaliranja od 0,5 kao rezultat jedinica bi postala 0,5 piksela i tako bi se oblici crtali u pola veličine. Na sličan način, postavljanje faktora skaliranja na 2.0 povećalo bi veličinu jedinice i jedna jedinica bi postala dva piksela. Zbog toga se crteži crtaju dvostruko veći. U ovom primjeru prikazat ćemo oblike s različitim faktorima skaliranja.

Primjer skaliranja:

```
function draw() {
  var ctx = document.getElementById('canvas').getContext('2d');

  // draw a simple rectangle, but scale it.
  ctx.save();
  ctx.scale(10, 3);
  ctx.fillRect(1, 10, 10, 10);
  ctx.restore();

  // mirror horizontally
  ctx.scale(-1, 1);
  ctx.font = '48px serif';
  ctx.fillText('MDN', -135, 120);
}
```

Slika 9. Prikaz koda za skaliranje

Rezultat koda:

Slika 10. Prikaz izvođenja koda za skaliranje

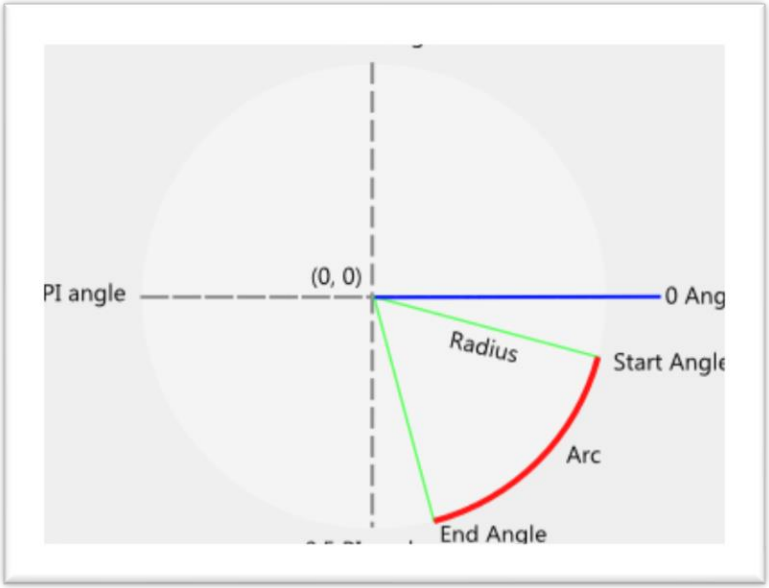
5. Oblici canvasa

Platno podržava samo jedan primitivni oblik - pravokutnike. Svi ostali oblici moraju se stvoriti kombiniranjem jedne ili više staza. Srećom, imamo kolekciju funkcija crtanja staza koje omogućuju sastavljanje vrlo složenih oblika.[22]

5.1. Crtanje kruga

Za stvaranje kruga ili luka na platnu, koristimo metodu arc (). Iako je stvaranje platna ravno na platnu s metodom lineTo () prilično jednostavno, metoda arc () uzima nekoliko dodatnih parametara. Krug je složenog oblika i da bismo definirali krug u koordinatama platna, potreban nam je položaj njegovog središta, tj. (x, y) i polumjera. Luk i krugovi vrlo su slični i tehnički nema razlike kad se crta ovo dvoje na platnu. Radi razumijevanja, lukove se mogu smatrati kao nepotpuni krug ili odjeljak kruga. Iako je krug potpuni okrugli oblik, luk može biti njegov dio.

Primjer: [12]



Slika 11. Grafički prikaz – presjek kruga

Luk prikazan na ovoj slici presjek je kruga. Da bismo definirali luk, moramo uzeti dva dodatna parametra - startAngle i endAngle. To su dvije točke na obodu zamišljenog kruga radi formiranja luka. Kao što je prikazano na dijagramu, na platnu html5, kutovi luka polaze od linije između središta i vodoravne crte u pravom smjeru (tj. Od 3 sata položaja sata). Posljednji važan parametar je smjer kuta od nulte kutne linije.

Arc() metoda ima sljedeće argumente: *arc (x, y, radius, stratAngle, endAngle, anticlockwise).*

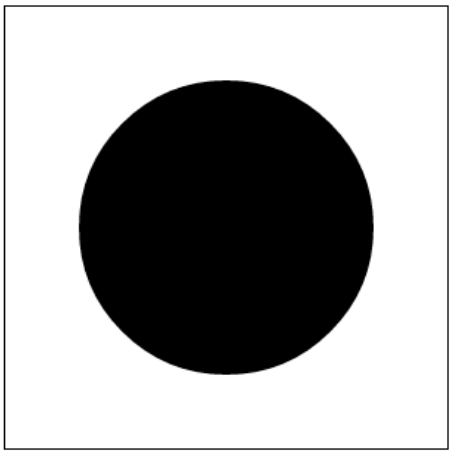
Kutovi luka (tj. StartAngle i endAngle) definirani su u radijanima, a ne u stupnjevima. Za stvaranje punog kruga (360 stupnjeva) potrebna su 2 PI radijana. Dakle, s 1 PI stvara se luk od pola kruga, a s 0,5 PI četvrtina kruga i tako dalje. Posljednji parametar tj. suprotno smjeru kazaljke na satu, nije obvezan i uzima booleove vrijednosti, tj. True i False. Ako se postavi na True, luk će se stvoriti u smjeru suprotnom od kazaljke na satu. Iako je ovaj parametar opcionalan, savjetuje se izričito definiranje False ili True jer će neki preglednici izbaciti pogrešku, ako je ne daju.

Primjer koda:[12]

```
<html>
<head>
    <script type="application/javascript">
        function drawShape() {
            var canvas = document.getElementById('canvas-tutorial');
            if (canvas.getContext) {
                var ctx = canvas.getContext('2d');
               ctx.beginPath();
                ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
               ctx.fill();
            }
        }
    </script>
    <style type="text/css">
        #canvas-tutorial
        {
            border: 1px solid #000000;
        }
    </style>
</head>
<body onload="drawShape();">
    <canvas id="canvas-tutorial" width="300" height="300">
"Sorry, your browser does not support canvas!"
</canvas>
</body>
</html>
```

Slika 12. Prikaz koda za crtanje kruga

Rezultat koda:



Slika 13. Prikaz izvođenja koda

Linija koja se koristi za definiranje kruga je:

ctx.arc (150, 150, 100, 0, 2 * Math.PI)

Ovdje određujemo središte kruga kao (150, 150). To je 150 piksela u pravom smjeru i 150 piksela prema dolje od gornjeg lijevog kuta platna. Krug ima polumjer od 100 piksela. Nakon toga definiramo početni i krajnji kut luka, koji je 0 i 2 PI. 2 PI jednaka je jednom cjelovitom krugu koji definira puni krug. Nakon toga postavljamo argument protiv kazaljke na satu na False. Tako se iz kuta 0 kreće kruženje pomicanjem krajnjeg kuta u smjeru kazaljke na satu. Ovdje nema nikakve razlike, jer crtamo kompletan krug. Međutim, ako nacrtamo luk, on će definirati na kojoj će strani kutne linije 0 biti luk.

5.2. Canvas putevi

Kao što sam već spomenuli, pravokutnici su jedini primitivni oblik na platnu. Za crtanje svih ostalih oblika trebamo koristiti platnene staze. A za korištenje staza moramo uključiti nekoliko dodatnih redaka koda.

- beginPath ()
- closePath ()
- stroke ()
- fill ()

Da bi se kreirao bilo koji novi oblik puta, zovemo metodu beginPath (). Dakle, ako se želi stvoriti više oblika unutar platna, svaki put se mora pozvati ova metoda. Sljedeće što se mora učiniti je pozvati moveTo () metodu. To postavlja početnu točku putanje koja će se stvoriti. Pretpostavimo da ćemo nacrtati oblik na stranici, prvo što trebamo je pomaknuti olovku ili olovku staviti u početni položaj. Ovdje radimo potpuno isto. Metoda moveTo uzima dva argumenta x i y kao koordinate početne točke. Nakon metode beginPath () i moveTo (), nacrtamo stvarne staze u obliku linija, lukova ili krivulja. Jednom kada završimo crtanje staza (linija ili lukova); pozivamo metodu closePath (). Na kraju, koristimo metode stroke () i fill () da definiramo želimo li samo obris ili popuniti nacrtane putove. [13]

5.2.1. Crtanje crte

Crtanje crte pomoću HTML5 Canvas je jednostavno, baš kao i crtanje crteža na papiru. Definira se put i zatim ispunjava taj isti put. Najosnovnija putanja koja se može nacrtati na platnu je ravna crta. Metode korištene u tu svrhu su moveTo (), lineTo () i stroke ().[14]

Koraci su:

Postavljanje trenutnog puta pomoću metode beginPath (). Premještanje kursor crtanja na početnu točku za stvaranje novog pod puta pomoću metode moveTo (x, y). Sljedeći korak je korištenje metode lineTo (x, y), koja dodaje novu točku i povezuje tu točku s početnom točkom koristeći ravnu liniju. Obje gore spomenute metode prihvaćaju x i y parametre koji točno govore gdje želimo povući crtu. Konačno, pomoću metode stroke () učinimo da linija bude vidljiva.

Sljedeći kod prikazuje kako nacrtati jednostavnu liniju od (10,45) do (180,40).

Primjer koda: [14]

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>Draw a line</title>
</head>
```

```
<body>
<canvas id="DemoCanvas" width="500" height="200"></canvas>
<script>
var canvas = document.getElementById('DemoCanvas');
if (canvas.getContext)
 {
  var context = canvas.getContext('2d');
  context.beginPath();
   context.moveTo(10,45);
  context.lineTo(180,47)
  context.stroke();
   }
</script>
</body>
</html>
```

Rezultat koda:

Slika 14. Prikaz izvođenja koda

5.2.2. Linije u boji

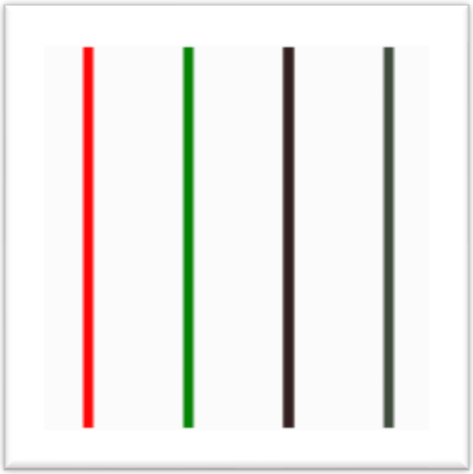
Za crtanje linija u boji možemo koristiti svojstvo strokeStyle, a zadana boja je crna. Sintaksa svojstva je objekt.strokeStyle = boja.

Primjer koda:[14]

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>Linije u boji</title>
</head>
<body>
<canvas id="DemoCanvas" width="400" height="400"></canvas>
<script>
var canvas = document.getElementById('DemoCanvas');
 if (canvas.getContext)
  {
```

```
    var context = canvas.getContext("2d");
     context.beginPath();
     context.moveTo(100, 10);
     context.lineTo(100, 200);
     context.lineWidth = 5;
     // postavljanje linija u boji
     context.strokeStyle = '#808000';
     context.stroke();
}
   </script>
     </body>
     </html>
```

Rezultat koda:



Slika 15. Prikaz izvođenja koda

5.3. Crtanje pravokutnika

Canvas podržava samo dva primitivna oblika: pravokutnike i staze (popisi točaka spojenih linijama). Svi ostali oblici moraju se stvoriti kombiniranjem jedne ili više staza. Srećom, imamo niz funkcija crtanja staza koje omogućuju sastavljanje vrlo složenih oblika.

Postoje tri funkcije koje crtaju pravokutnike na platnu:

- fillRect (x, y, širina, visina) - crta ispunjeni pravokutnik.
- strokeRect (x, y, širina, visina) - crta pravokutni obris.

- clearRect (x, y, širina, visina) - čisti točno pravokutno područje, čineći ga potpuno prozirnim.

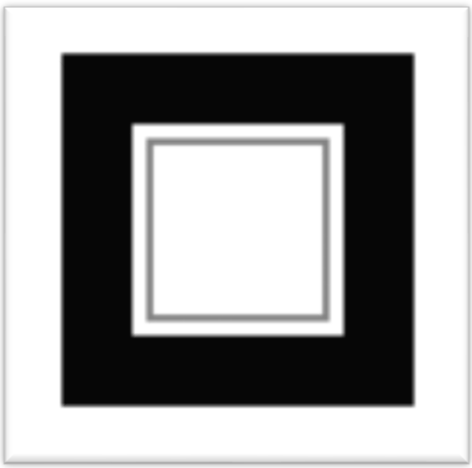
Svaka od ove tri funkcije uzima iste parametre, x i y određuju položaj na platnu (u odnosu na podrijetlo) gornjeg lijevog kuta pravokutnika. Širina i visina daju veličinu pravokutnika.

Primjer pravokutnika:

```
function draw() {
  var canvas = document.getElementById('canvas');
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d');

    ctx.fillRect(25, 25, 100, 100);
    ctx.clearRect(45, 45, 60, 60);
    ctx.strokeRect(50, 50, 50, 50);
  }
}
```

Slika 16. Prikaz koda za crtanje pravokutnika

Rezultat koda:



Slika 17. Prikaz izvođenja koda

Funkcija fillRect () crta veliki crni kvadrat od 100 piksela sa svake strane. Funkcija clearRect () briše kvadrat od 60x60 piksela iz središta, a onda strokeRect () poziva da stvori pravokutni obris 50x50 piksela unutar očišćenog kvadrata.[15]

5.4. Crtanje zvijezde

Sljedeći primjer stvara oblik zvijezde. [16]

```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 canvas star shape</title>
<script>
    function draw() {

        var canvas = document.getElementById('myCanvas');
        var ctx = canvas.getContext('2d');

        ctx.fillStyle = 'gray';

        var points = [ [ 0, 85 ], [ 75, 75 ], [ 100, 10 ], [ 125, 75 ],
            [ 200, 85 ], [ 150, 125 ], [ 160, 190 ], [ 100, 150 ],
            [ 40, 190 ], [ 50, 125 ], [ 0, 85 ] ];

        var len = points.length;

        ctx.beginPath();
        ctx.moveTo(points[0][0], points[0][1]);

        for (var i = 0; i < len; i++) {
            ctx.lineTo(points[i][0], points[i][1]);
        }

        ctx.fill();

    }
</script>
</head>

<body onload="draw();">
    <canvas id="myCanvas" width="350" height="250">
    </canvas>
</body>
</html>
```

Slika 18. Prikaz koda za crtanje zvijezde

Iz niza točaka stvara se oblik zvijezde.

```
var points = [ [ 0, 85 ], [ 75, 75 ], [ 100, 10 ], [ 125, 75 ],
    [ 200, 85 ], [ 150, 125 ], [ 160, 190 ], [ 100, 150 ],
    [ 40, 190 ], [ 50, 125 ], [ 0, 85 ] ];
```

To su koordinate zvijezde.

```
ctx.moveTo(points[0][0], points[0][1]);
```

Prelazimo na početnu koordinatu oblika pomoću metode moveTo ().

```
for (var i = 0; i < len; i++) {
    ctx.lineTo(points[i][0], points[i][1]);
}
```

Ovdje povezujemo sve koordinate zvijezde pomoću metode lineTo().

```
ctx.fill();
```

Metoda fill () ispunjava unutrašnjost oblika zvijezde definiranom (sivom) bojom.

Rezultat koda:

Slika 19. Prikaz izvođenja koda

6. Kvadratna i Bezierova krivulja [17]

Bezierove krivulje koriste se u računalnoj grafici za proizvodnju krivulja koje na svim mjerilima izgledaju prilično glatko. Izvorno ga je razvio Pierre Bézier u 1970-ima za CAD / CAM operacije. U vektorskoj grafici Bézierove krivulje koriste se za modeliranje glatkih krivulja koje se mogu skalirati u nedogled. "Staze", kako se obično nazivaju u programima za manipulaciju slike, kombinacija su povezanih Bézierovih krivulja. Bézierove krivulje također se koriste u animaciji kao alat za kontrolu kretanja. Bezier krivulje možemo crtati na isti način kao i crtanje linija, ali umjesto metode lineTo (), koriste se ili metoda bezierCurveTo () ili quadraticCurveTo (). Metoda bezierCurveTo () povezuje krajnje točke s kubičnom krivuljom bezier koristeći određeni par kontrolnih točaka, a metoda quadraticCurveTo () povezuje krajnje točke s kvadratnom bezierovom krivuljom koristeći jednu određenu kontrolnu točku.

6.1. Crtanje Bezier krivulje

Trebamo sljedeće metode kako bismo nacrtali Bezierove krivulje na platnu: [19]

- beginPath () - ova metoda resetira trenutni put.
- moveTo(x, y) - ovom metodom stvara se novi pod put s danom točkom.
- closePath() - ovom metodom trenutni pod put označava se zatvorenim i započinje novi pod put s točkom koja je jednaka početku i kraju novootvorenog pod puta.
- fill() - ova metoda ispunjava pod put trenutnim stilom ispunjavanja.
- stroke() - ova metoda pomiče pod puta s trenutnim stilom udaranja.
- bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y) - ovom metodom dana je točka trenutnoj stazi, a s prethodnom je povezana kubičnom Bezier krivuljom s danim kontrolnim točkama.

Kubična Bezierova krivulja mora sadržavati tri točke. Prve dvije točke (cp1x, cp1y) i (cp2x, cp2y) su kontrolne točke koje se koriste u kubnom Bézierovom proračunu, a zadnja točka (x, y) je krajnja točka krivulje.

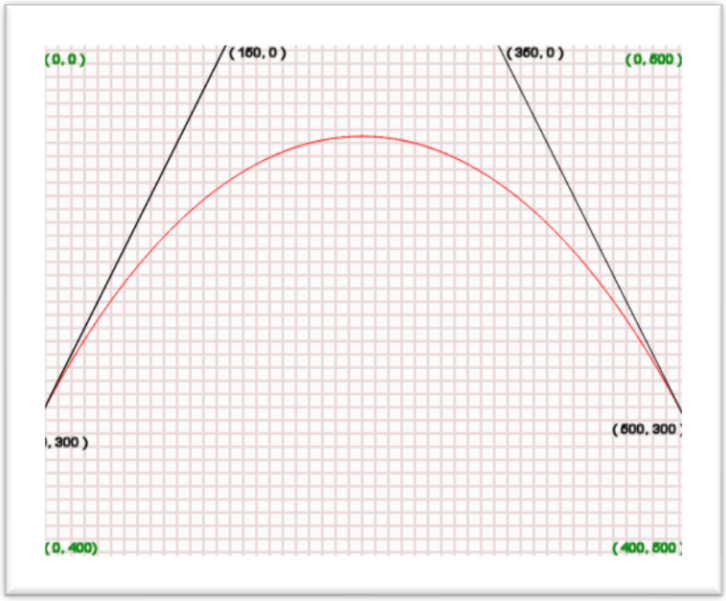
Sintaksa:

bezierCurveTo (cp1x, cp1y, cp2x, cp2y, x, y)

Parametri	**Tip**	**Opis**
cp1x	broj	x-koordinata prve Bézierove kontrolne točke
cp1y	broj	Y-koordinata prve Bézier-ove kontrolne točke
cp2x	broj	x-koordinata druge Bézier-ove kontrolne točke
cp2y	broj	Y-koordinata druge Bézierove kontrolne točke
x	broj	x-koordinata točke za dodavanje trenutnom putu
y	broj	Y-koordinata točke za dodavanje trenutnom putu

Prva točka na krivulji je posljednja točka postojećeg trenutnog pod puta. Ako put ne postoji, za stvaranje polazne točke upotrebljavaju se metode startPath i moveTo.

Primjer:



Slika 20. Grafički prikaz Bezier krivulje

U gornjem dijagramu:

(0, 0) je gornji lijevi položaj platna.

(0, 400) je pozicija lijevo dolje na platnu.

(0, 500) je gornja desna pozicija platna.

(400, 500) je položaj u donjem desnom platnu.

(0, 300) je polazna točka krivulje.

(150, 0), tj. (Cp1x, cp1y) je prvi kontrolni položaj krivulje.

(350, 0), tj. (Cp2x, cp2y) je drugi kontrolni položaj krivulje.

(500, 300), tj. (X, y) je krajnja točka krivulje.

U sljedećem primjeru vidjeti ćemo ravne linije i Bézierovu krivulju (između dvije točke) koja stvara znak 'R'.

Primjer koda:[19]

```
<!DOCTYPE html>
<html>
<head>
<title>Sample arcs example</title></head>
<body>
<canvas id="DemoCanvas" width="500" height="400"></canvas>
<script>
var canvas = document.getElementById("DemoCanvas");
var ctx = canvas.getContext("2d");
ctx.lineWidth = 7;
ctx.lineCap = "round";
ctx.clearRect(0, 0, canvas.width, canvas.height);
ctx.beginPath();
ctx.moveTo(30, 200);
ctx.lineTo(30, 50);
ctx.lineTo(65, 50);
ctx.moveTo(30, 120);
ctx.lineTo(65, 120);
ctx.lineTo(100, 200);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.beginPath();
ctx.moveTo(65, 50);
```

```
ctx.bezierCurveTo(120, 50, 120, 120, 65, 120);
ctx.strokeStyle = "green";
ctx.stroke();
</script>
</body>
</html>
```

Rezultat koda:

Slika 21. Prikaz izvođenja koda

6.2. Crtanje kvadratne krivulje

Kvadratna Bezierova krivulja zahtijeva dvije točke. Prva točka je kontrolna točka koja se koristi u kvadratnom Bézierovom proračunu, a druga točka je krajnja točka krivulje.

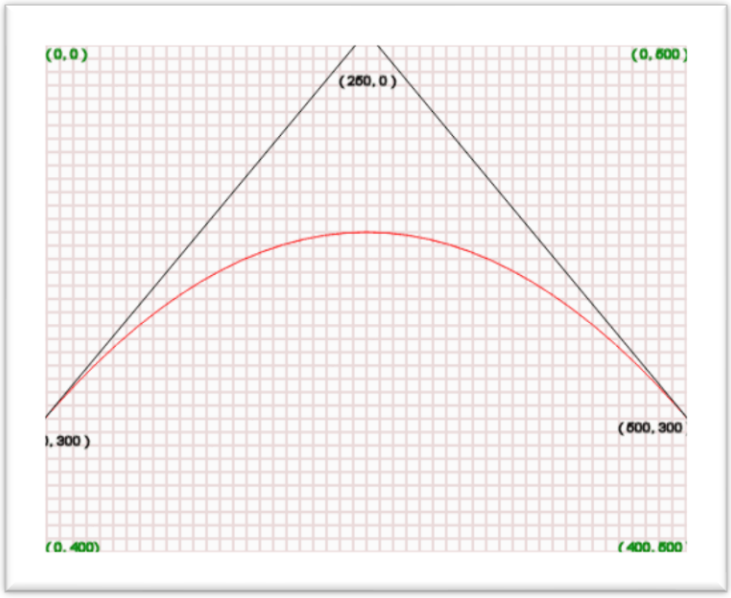
Sintaksa:

quadraticCurveTo (cp1x, cp1y, x, y);

Parametri	Tip	Opis
cp1x	broj	x-koordinata Bézier-ove kontrolne točke
cp1y	broj	y-koordinata Bézier-ove kontrolne točke
x	broj	x-koordinata točke za dodavanje trenutnom putu
y	broj	y-koordinata točke za dodavanje trenutnom putu

Polazna točka krivulje je posljednja točka postojećeg trenutnog pod puta. Ako put ne postoji, za početnu točku koristite se metode beginPath () i moveTo ().

Primjer:



Slika 22. Grafički prikaz kvadratne krivulje

U gornjem dijagramu:

- (0, 0) je gornji lijevi položaj platna.
- (0, 400) je pozicija lijevo dolje na platnu.
- (0, 500) je gornja desna pozicija platna.
- (400, 500) je položaj u donjem desnom platnu.
- (0, 300) je polazna točka krivulje.
- (250, 0), tj. (Cp1x, cp1y) je kontrolni položaj krivulje.
- (500, 300), tj. (X, y) je krajnja točka krivulje.

Primjer koda:

```
<!DOCTYPE html>
<html>
<head>
<title>Sample arcs example</title></head>
<body>
<canvas id="DemoCanvas" width="500" height="400"></canvas>
<script>
var canvas = document.getElementById('DemoCanvas');
  if (canvas.getContext)
```

```
  {
    var ctx = canvas.getContext('2d');
    // Crta oblike
    ctx.beginPath();
    ctx.moveTo(75,25);
    ctx.quadraticCurveTo(25,25,25,62.5);
    ctx.quadraticCurveTo(25,100,50,100);
    ctx.quadraticCurveTo(50,120,30,125);
    ctx.quadraticCurveTo(60,120,65,100);
    ctx.quadraticCurveTo(125,100,125,62.5);
    ctx.quadraticCurveTo(125,25,75,25);
    ctx.lineWidth = "3";
    ctx.strokeStyle = "green";
    ctx.stroke();
  } else {
    alert('Not supported in this browser.');
  }
</script>
</body>
</html>
```

Rezultat koda:

Slika 23. Prikaz izvođenja koda

7. Kinetic js

Eric Rowell, inženjer u tvrtki Yahoo stvorio je KineticJS - nevjerojatno moćnu JavaScript biblioteku HTML 5 Canvas koja proširuje 2D kontekst omogućujući interaktivnost canvasa za radne površine i mobilne aplikacije. Nažalost, koliko god je napredovala više se ne održava, ali još uvijek se može naći njena dokumentacija na

netu kao i verzije koje se onda mogu skinuti i koristiti, a posljednja stabilna verzija je kinetic - v5.1.0. Proći ćemo kroz neke njene osnovne značajke. Pomoću KineticJS-a možemo crtati oblike na pozornici i njima manipulirati pomoću sljedećih elemenata:

- Pomicanje
- Rotiranje
- Animiranje

Možemo crtati vlastite oblike ili slike pomoću postojećeg API-ja canvasa, premještati ih, skalirati i rotirati nezavisno od drugih oblika kako bi podržali animacije visokih performansi. Glavne prednosti KineticJS-a su sljedeće:

- Ubrzanje
- Skalabilnost
- Rastezljivost
- Fleksibilnost
- Poznavanje API-ja (za programere koji imaju znanje HTML, CSS, JS i jQuery)

Čak i ako aplikacija ima tisuće slika, animacija će se pokrenuti bez problema i s dovoljno visokim FPS-om (frames per second/sličica po sekundi). Predmeti su organizirani u slojeve, koje možemo imati onoliko koliko želimo. Oblici se također mogu organizirati u grupe. KineticJS omogućuje neograničeno gniježđenje oblika i grupa. Scene, slojevi, grupe i figure virtualni su čvorovi, slično DOM čvorovima u HTML-u. Bilo koji čvor može se oblikovati ili transformirati. Postoji nekoliko unaprijed definiranih oblika poput pravokutnika, krugova, slika, teksta, linija, poligona, zvijezda i slično. Također možemo izraditi prilagođene funkcije crtanja kako bi stvorili prilagođene oblike. Za svaki objekt možemo dodijeliti različite događaje (dodir ili miš). Na oblike možemo primijeniti filtar ili animaciju.

Naravno, sve potrebne funkcionalnosti HTML5 Canvasa možemo implementirati bez KineticJS, ali onda ćemo potrošiti puno više vremena, a ne nužno postići istu razinu performansi. Kreatori KineticJS-a stavili su svu svoju ljubav i vjeru u svjetliju budućnost u interaktivnost HTML5. Glavna prednost knjižnice je visoka izvedba, koja se postiže stvaranjem dva rendera na platnu. Jedan render je ono što se vidi, a drugi je posebno

skriveno platno koje se koristi za otkrivanje visokih performansi. Ogromna prednost KineticJS-a je što je to proširenje za HTML5 Canvas i stoga je savršeno pogodan za razvoj aplikacija za mobilne platforme. Visoke performanse mogu sakriti sve nedostatke platna u iOS-u, Androidu i drugim platformama. Poznata je činjenica da iOS platforma ne podržava Adobe Flash. U ovom slučaju, KineticJS je dobra Flash alternativa za iOS uređaje.

Ako želimo koristiti KineticJs trebamo napraviti sljedeću skriptu kao na slici ispod: [13]

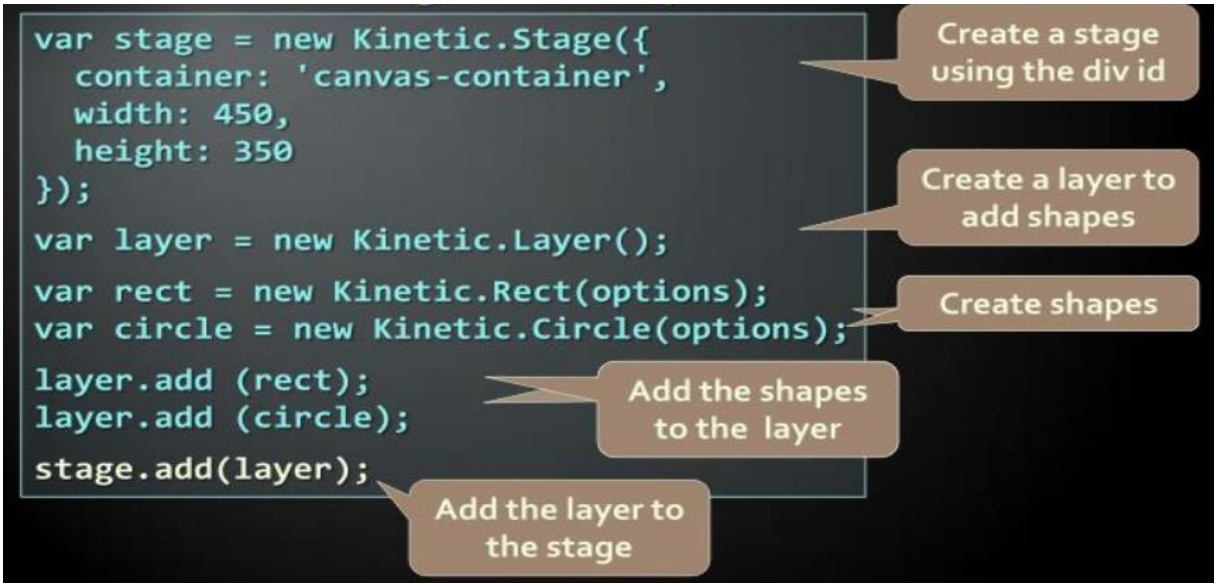
Create a stage using the div id – Stvoriti pozornicu koristeći div id

Create a layer to add shapes – Stvoriti sloj za dodavanje oblika

Create shapes – Stvoriti oblike

Add the shapes to the layers – Dodati oblike u slojeve

Add the layer to the stage - Dodati sloj pozornici

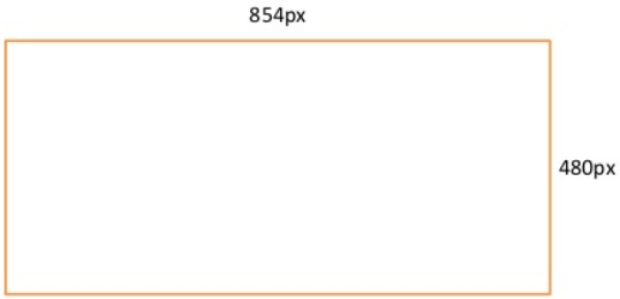


Slika 24. Prikaz koda za korištenje biblioteke KineticJs

[18] Četiri glavne komponente KineticJs-a: pozornica, slojevi, grupe i oblici

7.1. Pozornica

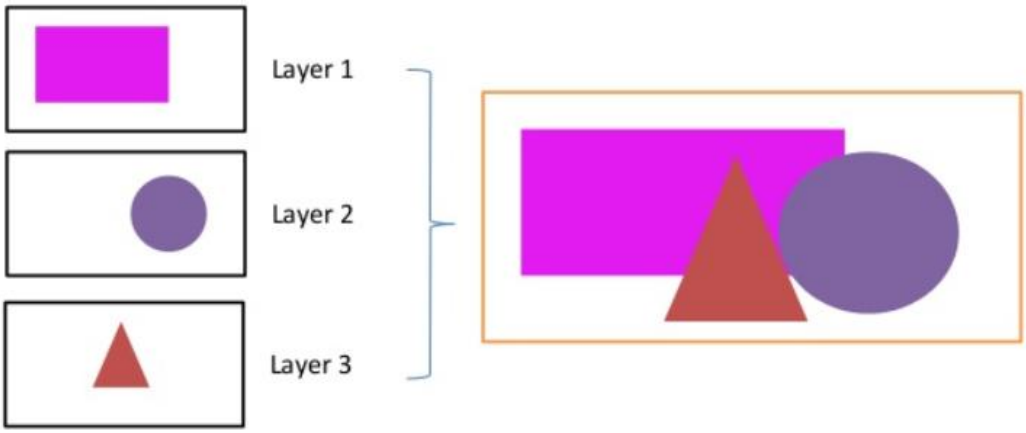
[14] KineticJS Stage je koji služi kao omotač oko jednog do mnogih KineticJS slojeva.



Slika 25. Prikaz pozornice

7.2. Slojevi

Slojevi su pojedinačni elementi HTML 5 platna unutar pozornice KineticJS složeni jedan preko drugog što pokazuje slika ispod.



Slika 26. Prikaz slojeva

[23] Uz pomoć slojeva prikladnije je nositi se s elementima. Elementi se mogu razdvojiti u slojeve, a može se postaviti i po jedan element na svaki sloj. Postavljanjem oblika ili

bilo kojeg drugog elementa u različite slojeve postići će učinak preklapanja jednog nad drugim. U KineticJS možet se stvoriti neograničen broj slojeva, ali preporučuje se upotreba dodatnih slojeva samo, ako su stvarno potrebni. To je zato što stvaranje svakog sloja u KineticJS uključuje stvaranje HTML5 Canvas elementa. Za primjenu je onda potrebno više resursa za obradu, poput memorije i vremena CPU-a. To je vrlo važno kod teških primjena. Ako imata puno animacija i filtera, također se preporučuje upotreba samo nekoliko slojeva, a ako je moguće i samo jedan. Pitanje koje se nameće je: "Kako se onda mogu stvoriti višeslojne aplikacije ako se preporučuje smanjenje broja slojeva?". KineticJS ima vrste virtualnih slojeva za svaki oblik. To znači da je svaki objekt u zasebnom virtualnom sloju, a dva različita elementa ne mogu se postaviti na istu razinu. Prema zadanom onaj koji se doda kasnije nalazi se na vrhu.

Za slojevitost se mogu koristiti sljedeće metode:

shape.moveToTop() - Ovo pomiče oblik na gornju razinu u sloju

shape.moveToBottom() - Ovo pomiče oblik na najnižu razinu u sloju

shape.moveUp() - Ovo pomiče oblik prema gore za jednu razinu

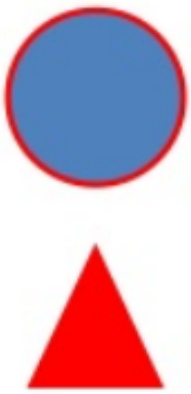
shape.moveDown() - Ovo pomiče oblik prema dolje za jednu razinu

shape.setZIndex(5) - Ovo pomiče oblik na razinu koju pruža z-indeks

Slojevi se mogu raditi ne samo s oblicima, već i s grupama. Sve prethodne metode mogu se upotrijebiti za oblike ili za skup oblika.

7.2. Oblici

U KineticJS postoji veliki broj unaprijed definiranih oblika. Ako se treba koristiti neki lik, prvo što bi se trebalo je potražiti da li se ta figura već nalazi na popisu standardnih oblika. Dostupni oblici su pravokutnik, kružnica, elipsa, klin, redoviti poligon, zvijezda, linija, poligon, slika, tekst, SVG staza, put teksta, pomak i mrlja.



Slika 27. Prikaz oblika

7.4. Grupe

[24] KineticJS grupe su zbirka KineticJS oblika ili drugih KineticJS grupa. Grupe su u osnovi kontejneri. Grupa je spremnik koji ima oblikovane predmete unutar slojeva. Ako se grupom manipulira, tada se elementi unutar te grupe također manipuliraju, ako povučemo grupu, tada se i njeni elementi povlače. Kad je objekt nacrtan u KineticJS on se zapravo crta dva puta.

Kako vidi korisnik:

Slika 28. Prikaz sa korisnikovog gledišta

Skriveno platno:

Slika 29. Prikaz skrivenog platna

7.5. Linije

[23] Linija je jedan od osnovnih pojmova geometrije. Postoje neka posebna svojstva za liniju u KineticJS.

Points - Ovo je niz piksela

lineCap - To tvori prijelom linije, koja može biti okrugla, kvadratna.

lineJoin - To tvori pridruživanje linija; može biti srednja, okrugla ili nagibna

dashArray - Ovo je niz linija podijeljenih u segmente

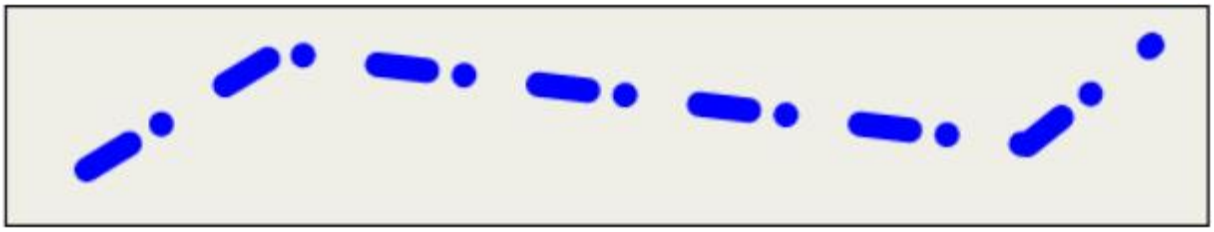
Primjer:

```
var blueLine = new Kinetic.Line({
    points: [73, 70, 150, 23, 450, 60, 500, 20],
    stroke: 'blue',
    strokeWidth: 10,
    lineCap: 'round',
    lineJoin: 'round',
    dashArray: [20, 15, 0.001, 30]
  });
```

Slika 30. Prikaz koda linije

U ovom se primjeru linija sastoji od četiri točke. Dvije od njih su početak i kraj, a ostale dvije savijaju crtu. Prijelomi linija i spojevi prikazani su zaobljenim završecima. dashArray je najzanimljivija značajka. Pomoću ovog svojstva mogu se linije vizualno

raščlaniti u segmente. Prvo se morata navesti duljinu segmenta, a zatim duljinu ureza do sljedećeg segmenta. Sljedeći segment može biti drugo područje s različitom duljinom i uvlačenjem. Čitav redak popunjava segmente opisane u dashArray.



Slika 31. Prikaz izvođenja koda

8. Zaključak

Element HTML5 <canvas> koji je izumio Apple, na kraju je dodan službenoj specifikaciji W3C HTML 5, postajući jedan od najzanimljivijih i najuzbudljivijih dijelova HTML 5. Koriste je web dizajneri diljem svijeta. HTML 5 upravo zbog svojih raznih značajki i mogućnosti koje pruža u zadnje vrijeme i brzini razvijanja, ključan je u izradi web stranica jer korisniku pruža paletu karakteristika i mogućnosti koji se mogu koristiti. Canvas 2D API nudi moćne grafičke funkcionalnosti koji se mogu koristiti za implementaciju kreativnih i uvjerljivih aplikacija koje se pokreću u skoro svakom novijem pregledniku, a nisu potrebni nikakvi priključci (dodatci). Canvas će u budućnosti sigurno još više napredovati i postati neizostavan element u HTML 5 za izradu animacija, 2D/3D grafike, igara i respozivnih, interaktivnih web stranica.

9. Literatura

1. http://www.webtech.com.hr/html5.php (01.09.2019)
2. http://www.webtech.com.hr/html5.php(01.09.2019)
3. https://www.w3resource.com/html5/introduction.php (01.09.2019)
4. https://www.w3schools.com/html/html5_intro.asp (01.09.2019)
5. https://developer.mozilla.org/hr/docs/Web/Guide/HTML/HTML5 (05.09.2019)
6. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial (05.09.2019)
7. https://www.w3schools.com/graphics/canvas_intro.asp(05.09.2019)
8. https://www.geeksforgeeks.org/html5-introduction/(03.09.2019)
9. https://www.w3resource.com/html5-canvas/(03.09.2019)
10. https://blog.logrocket.com/when-to-use-html5s-canvas-ce992b100ee8/(03.09.2019)
11. http://tutorials.jenkov.com/html5-canvas/transformation.html(03.09.2019)
12. https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/translate(04.09.2019)
13. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Transformations(04.09.2019)
14. http://www.webtutorialplus.com/html5-canvas-tutorial-circles-arcs-and-curves/(10.09.2019)
15. http://www.webtutorialplus.com/html5-canvas-shapes/(10.09.2019)
16. https://www.tutorialrepublic.com/html-tutorial/html5-canvas.php(10.09.2019)
17. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_shapes(14.09.2019)
18. http://zetcode.com/gfx/html5canvas/shapes/(14.09.2019)
19. https://www.slideserve.com/vinnie/kineticjs (14.09.2019)
20. https://www.slideshare.net/VictorMichnowicz/html-5-with-kineticjs(16.09.2019)
21. https://www.tutorialspoint.com/html5/canvas_drawing_bezier.htm(16.09.2019)
22. https://webplatform.github.io/docs/tutorials/canvas/Canvas_tutorial/Drawing_shapes/(16.09.2019)
23. https://subscription.packtpub.com/book/web_development/9781849699433/1/ch00lvl1sec06/top-features-you-need-to-know-about?fbclid=IwAR1MBzPBv9ixVAFmx7aAdn3ZGQJYshgzTP9oU1ytd1EH7a9XvdKz14LHvW8
24. https://www.tutorialspoint.com/What-are-the-differences-between-group-and-layer-in-KineticJs-with-HTML

Sažetak

Naslov: Interaktivni grafički objekti u HTML 5 i KineticJs

U radu su opisane mogućnosti koje pruža HTML 5 kao i korištenje elementa <canvas> koji se može koristiti za prikazivanje grafike na internetu, stvaranje 2D oblika i bitmap slika. Prikazani su osnovni primjeri crtanja 2D objekata u canvasu kao što su, transformacije, oblici, putevi, krivulje i ostali objekti. Opisana je i biblioteka KineticJs koja proširuje 2D kontekst omogućujući interaktivnost canvasa za radne površine i mobilne aplikacije. Prikazane su neke njene osnovne značajke i komponente kao što su pozornica, slojevi, grupe i oblici.

Ključne riječi: HTML 5, Canvas, Canvas API, KineticJs, Računalna grafika

Abstract

Title: Interactive Graphic Objects in HTML 5 and KineticJs

In this work it is described the capabilities of HTML 5 as well as the use of <canvas> elements that can be used to display graphics on the internet, create 2D shapes and bitmap images. Basic examples of drawing 2D objects in canvas are shown, such as, transformations, shapes, paths, curves, and other objects. Also described is a KineticJs library that extends the 2D context by enabling canvas for desktop and mobile applications. There are presented some of basic features and components such as stages, layers, groups, and shapes.

Ključne riječi: HTML 5, Canvas, Canvas API, KineticJs, Računalna grafika