Crtanje geometrijskih oblika

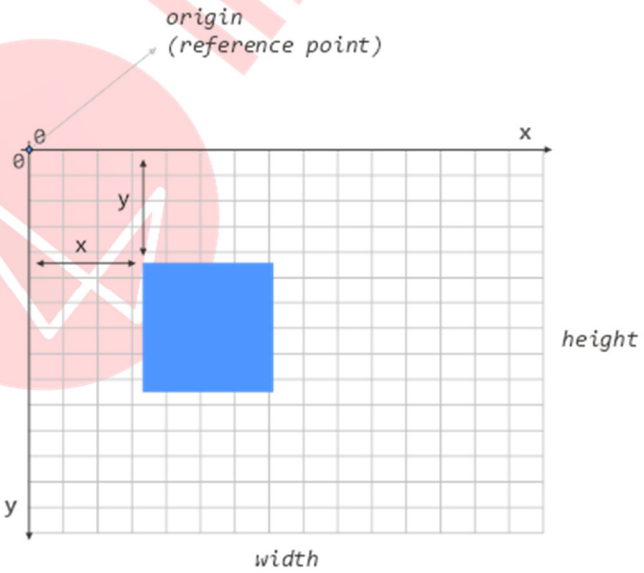
U prethodnoj lekciji uspostavljeno je okruženje za obavljanje crtanja grafike unutar canvas elementa korišćenjem Canvas API-a. U lekciji koja je pred Vama, nastavljamo tamo gde smo se zaustavili u prethodnoj, pa ćete tako imati prilike da naučite osnovne pristupe za crtanje grafike unutar canvas elementa korišćenjem Canvas API-a. Biće prikazani osnovni postulati crtanja proizvoljne grafike, korišćenjem brojnih ugrađenih metoda Canvas API-a.

Koordinatni sistem canvas elementa

U prethodnoj lekciji obavljeno je crtanje prvog oblika unutar canvas-a. Bio je to jedan kvadrat, čije crtanje je postignuto na sledeći način:

```
var ctx = myCanvas.getContext('2d');

ctx.fillStyle = 'rgb(79, 149, 255)';
ctx.c(20, 20, 100, 100);
```

Crtanje bilo koje grafike unutar canvas-a, podrazumeva poznavanje osobina koordinatnog sistema canvas elementa. To možete videti i na prvom primeru crtanja kvadrata. Za razumevanje parametara koji se prosleđuju metodi `fillRect()` neophodno je poznavati osobine spomenutog koordinatnog sistema. Kako nacrtani kvadrat izgleda unutar canvas koordinatnog sistema, možete videti na slici 5.1.



Slika 5.1 - Koordinatni sistem canvas elementa

Sa slike 5.1 možete da vidite da se referenta tačka koordinatnog sistema canvas elementa nalazi u gornjem, levom uglu. Takva tačka ima koordinate (0,0) i predstavlja početak koordinatnog sistema. Ova tačka se naziva referentna, zato što se u odnosu na nju obavlja crtanje grafike unutar canvas elementa. To možete videti i na primeru crtanja kvadrata. Prva dva parametra metode `fillRect()`, odnosila su se na udaljenost kvadrata od leve i gornje ivice, respektivno.

Pitanje

Referentna tačka koordinatnog sistema ima koordinate:

a) (0,0)
b) (-1,-1)
c) (10,10)
d) (1,1)

Objašnjenje:

Referenta tačka koordinatnog sistema canvas elementa ima koordinate (0, 0) i predstavlja početak koordinatnog sistema.

Crtanje osnovnih oblika

Canvas API poseduje samo dva ugrađena mehanizma za crtanje geometrijskih oblika unutar canvas-a. Oni direktno omogućavaju crtanje pravougaonika i putanja (*paths, engl.*). Svi ostali geometrijski oblici moraju se kreirati korišćenjem sistema za crtanje dva upravo spomenuta tipa oblika.

Crtanje pravougaonika i kvadrata

Crtanje pravougaonika se može obaviti korišćenjem nekoliko metoda, prikazanih tabelom 5.1.

Metoda	Opis
`fillRect(x, y, width, height)`	za crtanje pravougaonika sa ispunom
`strokeRect(x, y, width, height)`	za crtanje pravougaonika bez ispune, odnosno samo sa okvirima
`clearRect(x, y, width, height)`	za brisanje pravougaonog prostora unutar canvas-a; brisanje podrazumeva postavljanje boje svih piksela takvog pravougaonika na transparentu crnu

Tabela 5.1 - Metode za crtanje pravougaonika

Iz tabele 5.1 možete videti da sve metode za crtanje pravougaonika prihvataju identičan skup parametara:

- `x` - x koordinata referentne tačke pravougaonika, odnosno udaljenost leve ivice pravougaonika od leve ivice `canvas`-a
- `y` - y koordinata referentne tačke pravougaonika, odnosno udaljenost gornje ivice pravougaonika od gornje ivice `canvas`-a
- `width` - širina pravougaonika
- `height` - visina pravougaonika

Napomena

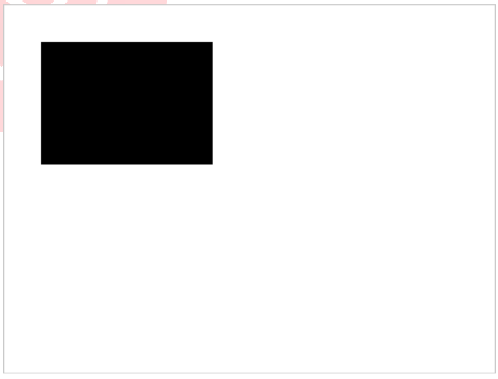
Šablon koji je kreiran u prethodnoj lekciji, biće korišćen i u ovoj. Stoga je kod koji bude prikazan u nastavku, potrebno postavljati unutar `draw()` funkcije, nakon naredbe za dobijanje prostora za crtanje:

```
function draw() {
    let myCanvas = document.getElementById("my-canvas");

    if (myCanvas.getContext) {
        var ctx = myCanvas.getContext('2d');

        //code goes here

    } else {
        alert("Canvas is not supported.");
    }
}
```

Naredba za crtanje jednog pravougaonika može da izgleda ovako:

```
ctx.fillRect(30, 30, 140, 100);
```

Ovo je naredba koja je već upotrebljena u prethodnoj lekciji. Reč je o naredbi kojom se poziva metoda za crtanje pravougaonika sa ispunom. Ipak, ovoga puta se ova metoda poziva bez bilo kakvog prethodnog definisanja boje, pa se unutar `canvas`-a dobija pravougaonik kao na slici 5.2.



Slika 5.2 - Pravougaonik sa ispunom

Sa slike 5.2 možete videti da bez prethodnog definisanja boje, pravougaonik dobija crnu boju.

Pravougaonik bez ispune se može nacrtati na sledeći način:

```
ctx.strokeRect(30, 30, 140, 100);
```

Sada se unutar canvas-a dobija pravougaonik kao na slici 5.3.

Slika 5.3 - Pravougaonik bez ispune

Boja pozadine, stil i boja okvira

Upravo nacrti pravougaonici poseduju podrazumevanu stilizaciju. Stilizacija se odnosi na boju ispune i boju i stil okvira. Kako uticati na stilizaciju, biće prikazano u narednoj lekciji, nakon upoznavanja svih pristupa za crtanje različitih geometrijskih oblika.

Još jedna metoda koja pripada skupu funkcionalnosti za crtanje pravougaonika jeste metoda koja omogućava da se određeni pravougaoni prostor unutar canvas-a obriše. Brisanje podrazumeva postavljanje providnosti za sve piksele koji pripadaju prostoru definisanog pravougaonika:

```
ctx.fillRect(30, 30, 140, 100);
ctx.clearRect(85, 65, 85, 65);
```

Sada je prvo obavljeno crtanje jednog pravougaonika sa ispunom. Zatim je drugom naredbom deo tako nacrtanog pravougaonika obrisan, pa se na kraju dobija efekat kao na slici 5.4.

Slika 5.4 - Pravougaonik čiji je jedan deo obrisan korišćenjem metode clearRect()

Putanje

Do sada je prikazano kako se unutar `canvas`-a crtaju pravougaonici. Crtanje svih ostalih geometrijskih oblika (krugovi, linije, rombovi, paralelogrami, trouglovi…) obavlja se korišćenjem putanja. Korišćenjem putanja moguće je nacrtati čak i pravougaonike i kvadrate.

Putanje su sačinjene iz tačaka koje su međusobno povezane linijama, pri čemu takve linije mogu biti prave, ali i krive. Tako je korišćenjem putanja moguće nacrtati bilo koji geometrijski oblik.

Kreiranje, odnosno crtanje putanja podrazumeva sprovođenje tri osnovna koraka:

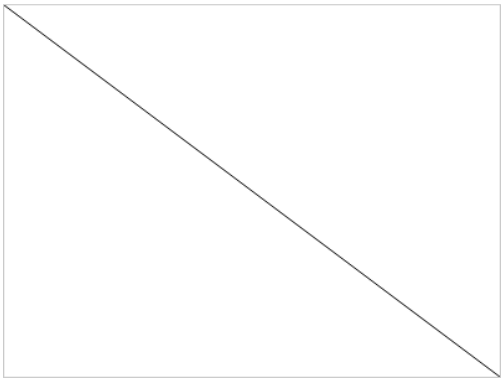
1. kreiranje putanje korišćenjem metode `beginPath()`
2. definisanje početne tačke putanje upotrebom metode `moveTo()`
3. korišćenje komandi za crtanje kojima se definišu osobine putanje; postoje brojne metode kojima je moguće definisati osobine putanje koja se crta
4. crtanje putanje pozivanjem metoda `stroke()` ili `fill()`

Crtanje pravih linija

Prvi primer rada sa putanjama ilustrovaće crtanje jedne prave linije. Na tom primeru videćete kako se praktično sprovode sva četiri upravo navedena koraka:

```
ctx.beginPath();
ctx.moveTo(0, 0);
ctx.lineTo(400, 300);
ctx.stroke();
```

Ovakvim kodom, unutar canvas elementa dobija se linija kao na slici 5.5.



Slika 5.5 - Putanja kojom se predstavlja jedna linija unutar canvas-a

Primer ilustruje sva četiri nešto ranije navedena koraka. Prvo se kreira nova putanja korišćenjem metode `beginPath()`. Zatim se postavlja početna tačka putanje korišćenjem metode `moveTo()`. Za početnu tačku je definisana referentna tačka koordinatnog sistema, sa koordinatama `(0,0)`. To praktično znači da će crtanje putanje da započne iz gornjeg, levog ugla canvas elementa.

Metoda moveTo()

Kreiranje putanje korišćenjem Canvas API-a, najviše podseća na crtanje olovkom na papiru. Po takvoj logici, metoda `moveTo()` simulira podizanje olovke od papira i njeno premeštanje na neku drugu poziciju. Tako metoda `moveTo()` ne obavlja nikakvo konkretno crtanje, već omogućava da se imaginarna olovka za crtanje postavi na neku konkretnu poziciju unutar canvas elementa.

Metoda `moveTo()` prihvata dva ulazna parametra:

```
moveTo(x, y)
```

Parametri metode `moveTo()` odnose se na koordinate tačke na koju će da bude pozicionirana *olovka* za crtanje.

U upravo prikazanom primeru, osobine putanje koja se crta, definišu se metodom `lineTo()`.

Metoda lineTo()

Crtanje pravih linija prilikom konstruisanja putanje, obavlja se korišćenjem metode `lineTo()`. Ova metoda obavlja crtanje prave linije, spajanjem trenutne tačke na kojoj se unutar canvas-a nalazi imaginarna olovka za crtanje i tačke čije se koordinate ovoj metodi prosleđuju kao parametri:

```
lineTo(x, y)
```

Parametri `x` i `y` odnose se na koordinate završne tačke linije koja će biti dodata putanji koja se crta.

Na kraju, kako bi se kreirana putanja nacrtala unutar canvas elementa, neophodno je pozvati jednu od metoda za crtanje i to u zavisnosti od efekata koji se želi postići. U prikazanom primeru crtanja prave linije, upotrebljena je metoda `stroke()`.

Metoda stroke()

Metoda `stroke()` koristi se za crtanje prethodno kreirane putanje unutar canvas elementa. Pri tom se crtaju samo okviri eventualnog oblika koji se dobija kreiranom putanjom.

Crtanje isprekidanih linija

Korišćenjem već prikazanih metoda Canvas API-a, sada možemo stvoriti jednu isprekidanu liniju. Kod će izgledati ovako:

```
ctx.beginPath();
ctx.moveTo(0, 150);
ctx.lineTo(25, 150);
ctx.moveTo(50, 150);
ctx.lineTo(75, 150);
ctx.moveTo(100, 150);
ctx.lineTo(125, 150);
ctx.moveTo(150, 150);
ctx.lineTo(175, 150);
ctx.moveTo(200, 150);
ctx.lineTo(225, 150);
ctx.moveTo(250, 150);
ctx.lineTo(275, 150);
ctx.moveTo(300, 150);
ctx.lineTo(325, 150);
ctx.moveTo(350, 150);
ctx.lineTo(375, 150);
ctx.moveTo(400, 150);
ctx.stroke();
```

Ovakvim kodom dobija se isprekidana linija kao na slici 5.6.

Slika 5.6 - Isprekidana linija unutar canvas elementa

Upravo prikazani primer ilustrovao je kombinaciju metoda `moveTo()` i `lineTo()` kako bi se dobila isprekidana linija. Prazni prostori unutar putanje dobijeni su pomeranjem imaginarne olovke za crtanje korišćenjem metode `moveTo()`.

Stilizacija nacrtane grafike

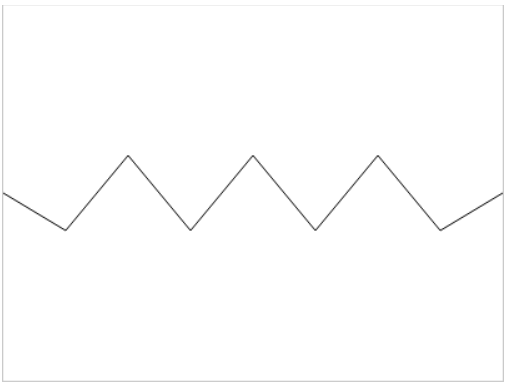
Upravo prikazani primer moguće je postići na mnogo jednostavniji način, upotrebom pristupa za stilizovanje nacrtane grafike. O tome će biti više reči u nastavku ove priče o crtanju korišćenjem Canvas API-a.

Crtanje izlomljenih linija

Izlomljene linije se mogu dobiti nadovezivanjem uzastopnih poziva `lineTo()` metode:

```
ctx.beginPath();
ctx.moveTo(0, 150);
ctx.lineTo(50, 180);
ctx.lineTo(100, 120);
ctx.lineTo(150, 180);
ctx.lineTo(200, 120);
ctx.lineTo(250, 180);
ctx.lineTo(300, 120);
ctx.lineTo(350, 180);
ctx.lineTo(400, 150);
ctx.stroke();
```

Na ovaj način, unutar `canvas` elementa se dobija linija kao na slici 5.7.



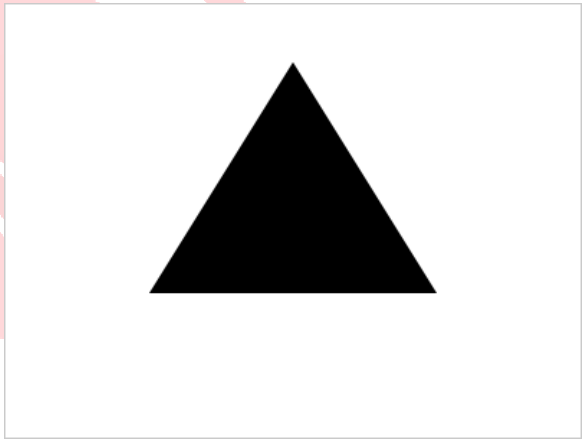
Slika 5.7 - Izlomljena linija unutar canvas elementa

Crtanje trouglova

Dosadašnji primeru su podrazumevali crtanje putanja kod kojih početna i završna tačka nisu bile spojene. Spajanjem prve i poslednje tačke putanje, stvara se mogućnost za dobijanje različitih geometrijskih oblika koji poseduju ispunu, pa je tako moguće predstaviti trougao (ili bilo koji višeugao), krug, elipsu i slično. Prvo će biti ilustrovao crtanje jednog trougla:

```
ctx.beginPath();
ctx.moveTo(200, 40);
ctx.lineTo(300, 200);
ctx.lineTo(100, 200);
ctx.fill();
```

Ovakvim kodom, dobija se trougao kao na slici 5.8.



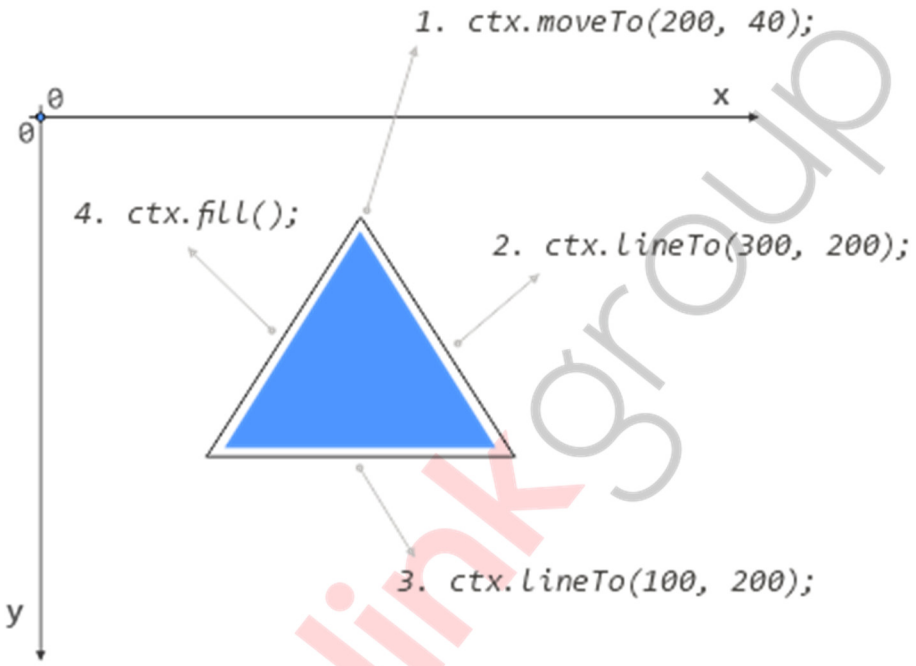
Slika 5.8 - Trougao unutar canvas elementa

Crtanje trougla postignuto je crtanjem jedne izlomljene linije, koja predstavlja desnu i donju ivicu trougla. Ipak, bitno je da primetite da na taj način dobijena putanja nije u potpunosti zatvorena, zato što nije obavljeno povezivanje prve i poslednje tačke putanje. Tako nešto za nas je obavila metoda `fill()`.

Metoda fill()

Metoda `fill()` koristi se za crtanje oblika koji se dobijaju ispunjavanjem površine koja je ograničena jednom putanjom. Pri tom, metoda `fill()` za nas automatski obavlja zatvaranje kreirane putanje, spajanjem njene prve i poslednje tačke.

Kompletna logika crtanja trougla iz prikazanog primera ilustrovana je slikom 5.9.



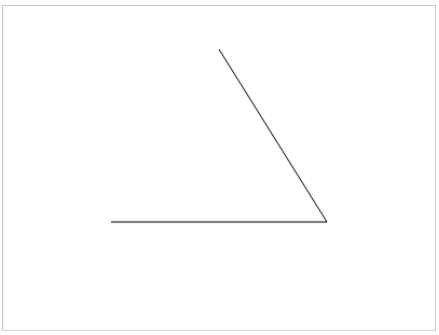
Slika 5.9 - Logika po kojoj se obavlja crtanje trougla

Na slici 5.9 možete videti korake koji su sprovedeni prilikom crtanja trougla, kao i njihov redosled.

Veoma je bitno reći da se automatsko zatvaranje putanje obavlja isključivo od strane metode `fill()`, ali ne i od strane metode `stroke()` koja je ilustrovana nešto ranije u ovoj lekciji. Drugim rečima, kod koji će biti prikazan u nastavku neće da proizvede trougao bez ispune:

```
ctx.beginPath();
ctx.moveTo(200, 40);
ctx.lineTo(300, 200);
ctx.lineTo(100, 200);
ctx.stroke();
```

Jedina razlika se odnosi na poziv metode `stroke()` umesto metode `fill()`. Efekat je kao na slici 5.10.



Slika 5.10 - Metoda stroke() ne obavlja automatsko zatvaranje kreirane putanje

Kako bi se dobio željeni efekat, odnosno trougao bez ispune, neophodno je eksplicitno zatvoriti putanju. To se postiže korišćenjem metode `closePath()`.

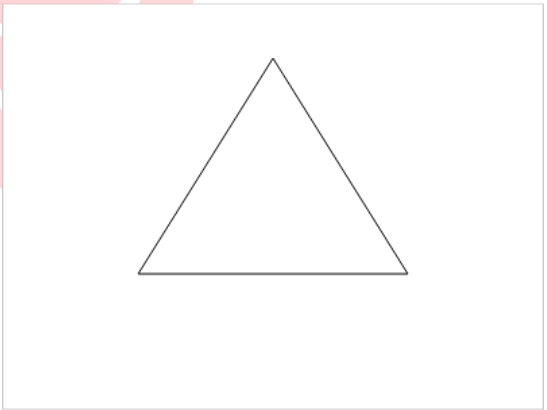
Metoda closePath()

Metoda `closePath()` obavlja zatvaranje putanje, tako što dodaje jednu pravu liniju koja spaja poslednju i prvu tačku putanje.

Metoda `closePath()` se u našem primeru može postaviti pre poziva metode `stroke()`:

```
ctx.beginPath();
ctx.moveTo(200, 40);
ctx.lineTo(300, 200);
ctx.lineTo(100, 200);
ctx.closePath();
ctx.stroke();
```

Na ovaj način će zatvaranje putanje biti eksplicitno obavljeno, pa će se dobiti efekat kao na slici 5.11.



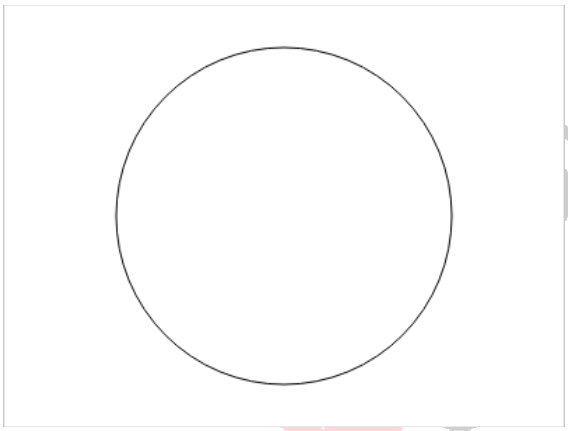
Slika 5.11 - Trougao bez ispune

Crtanje kružnih lukova, kružnica i krugova

Crtanje kružnih lukova, pa samim tim i kružnica i krugova, može se postići korišćenjem metode `arc()`. Sa osobinama ove metode, upoznaćemo se na primeru crtanja jedne kružnice:

```
ctx.beginPath();
ctx.arc(200, 150, 120, 0, 2 * Math.PI);
ctx.stroke();
```

Na ovaj način dobija se kružnica kao na slici 5.12.



Slika 5.12 - Kružnica

Način na koji je dobijen ovakav prikaz zahteva razumevanje parametara metode `arc()`.

Metoda arc()

Metoda `arc()` koristi se za crtanje kružnih lukova. Za obavljanje takvog posla, ona prihvata nekoliko parametara:

```
arc(x, y, radius, startAngle, endAngle, anticlockwise)
```

Uloga navedenih parametara je sledeća:

- `x` - x koordinata centra
- `y` - y koordinata centra
- `radius` – poluprečnik
- `startAngle` - vrednost ugla u radijanima na kome se nalazi početna tačka kružnog luka; merenje ugla započinje od početka x ose, ka njenoj pozitivnoj strani (u smeru kazaljke na satu)
- `endAngle` - vrednost ugla u radijanima na kome se nalazi krajnja tačka kružnog luka
- `anticlockwise` - usmerenje kružnice; kada se za vrednost postavi `true`, kružni luk se crta u smeru obrnutom smeru kazaljke na satu; ovaj parametar nije obavezan i poseduje podrazumevanu vrednost `false`

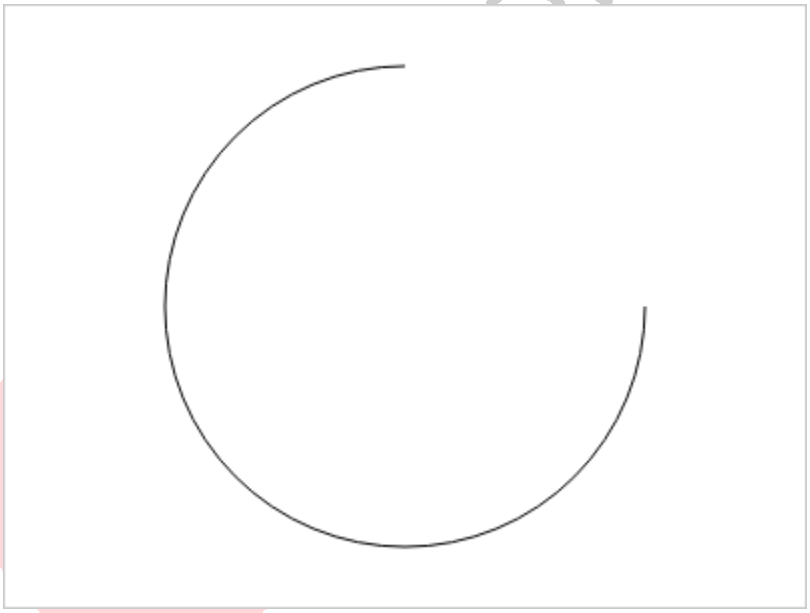
Nakon upoznavanja parametara metode `arc()` lakše je razumeti kako je u prethodnom primeru obavljeno crtanje kruga unutar `canvas` elementa. Centar kruga je postavljen u sam centar `canvas` elementa, definisanjem vrednosti `200` i `150`, za `x` i y koordinate, respektivno.

Trećim parametrom, poluprečnik kruga je postavljen na 120, što praktično znači da će krug biti širok 240px.

Na kraju, definisan je i početak i kraj kružnog luka. Za početak kružnog luka je postavljena vrednost 0, dok je za njegovu poslednju tačku definisana ona koja se nalazi na uglu 2π (`2 * Math.PI`). To je zapravo jedna ista tačka, koja se nalazi na 0, odnosno 360 stepeni, kada se upravo navedene vrednosti iz radijana prevedu u stepene. Zbog odabira ovakvih uglova, u primeru se dobija puna kružnica, a ne samo njen isečak. Ukoliko je potrebno predstaviti isečak kružnice, odnosno kružni luk, potrebno je definisati drugačije uglove:

```
ctx.beginPath();
ctx.arc(200, 150, 120, 0, 3 * Math.PI/2);
ctx.stroke();
```

Malom izmenom nad poslednjim parametrom, dobija se samo deo kružnice, odnosno kružni luk (slika 5.13).

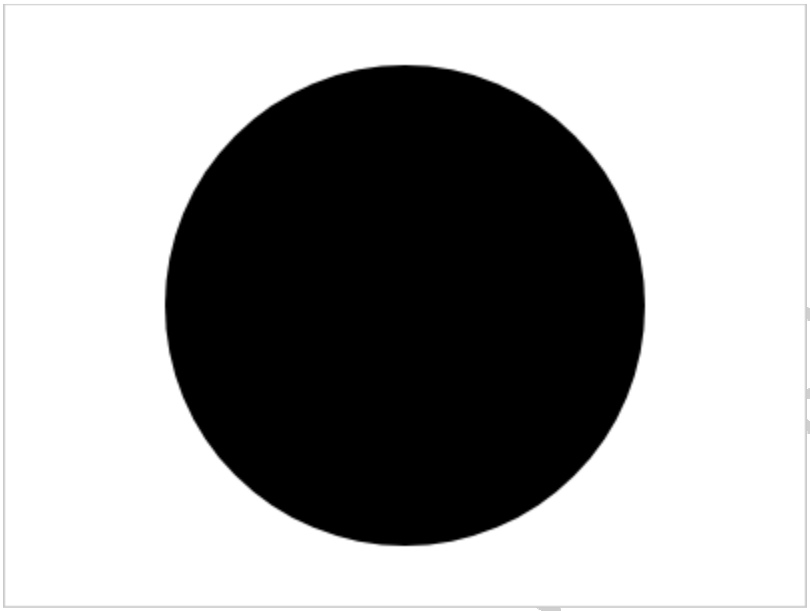


Slika 5.13 - Kružni luk

Prikazani primeri su podrazumevali crtanje kružnica i kružnih lukova. Ukoliko je potrebno nacrtati krug, dovoljno je konstruisati kružnicu kao na slici 5.12 i umesto metode `stroke()`, upotrebiti metodu `fill()`:

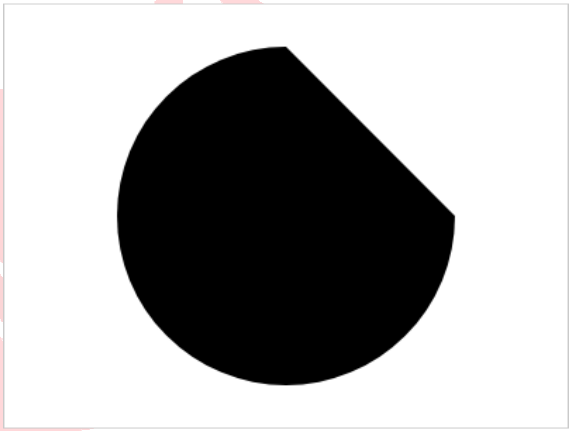
```
ctx.beginPath();
ctx.arc(200, 150, 120, 0, 2 * Math.PI);
ctx.fill();
```

Efekat je kao na slici 5.14.



Slika 5.14 - Krug

Ipak, kada se metoda `fill()` upotrebi nakon konstruisanja kružnog luka, kao na slici 5.13, obavlja se automatsko zatvaranje putanje, ali jednom pravom linijom, pa se tako dobija efekat kao na slici 5.15.



Slika 5.15 - Krug kome nedostaje jedan deo

Ovakav efekat je dobijen sledećim kodom:

```
ctx.beginPath();
ctx.arc(200, 150, 120, 0, 3 * Math.PI/2);
ctx.stroke();
```

Crtanje pravougaonika

Na kraju, pomoću putanja je moguće crtati i pravougaonike. Za obavljanje takvog posla se koristi metoda `rect()`.

Metoda rect()

Metoda `rect()` se koristi za dodavanje pravougaonika putanji koja se crta. Metoda poseduje sledeći oblik:

```
rect(x, y, width, height)
```

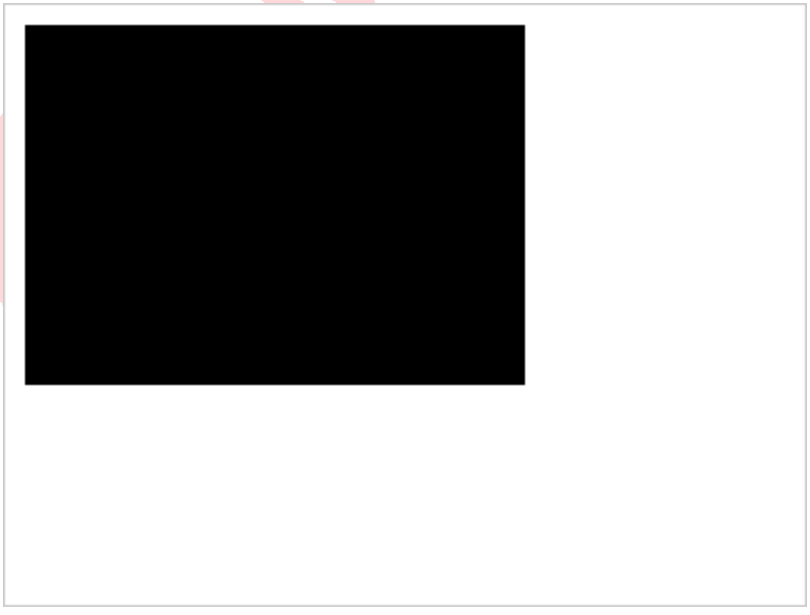
Parametri imaju sledeće značenje:

- `x` - x koordinata referentne tačke pravougaonika
- `y` - y koordinata referentne tačke pravougaonika
- `width` - širina pravougaonika
- `height` - visina pravougaonika

Primer crtanja pravougaonika korišćenjem `rect()` metode može da izgleda ovako:

```
ctx.beginPath();
ctx.rect(10, 10, 250, 180);
ctx.fill();
```

Na ovaj način se dobija pravougaonik kao na slici 5.16.



Slika 5.16 - Pravougaonik nacrtan kao putanja sa ispunom

Primer - Programabilno crtanje kružnog grafikona (Pie Chart)

Primer koji će uslediti. pokušaće da sumira veći deo onoga što je obrađeno u lekciji za nama. Primer će ilustrovati programabilno, dinamičko crtanje kružnog dijagrama. Ideja je da korisnik bude u mogućnosti da definiše proizvoljan broj delova u procentima i da se zatim na osnovu takvih podataka obavi crtanje kružnog dijagrama. Evo kako će izgledati kod koji obavlja opisano:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Canvas Drawing</title>
    <style>
        #my-canvas {
            border: #cacaca 1px solid;
        }
    </style>
</head>
<body>
    <canvas id="my-canvas" width="400" height="300">
        Your web browser does not support canvas element.
    </canvas>
    <script>
        window.onload = draw;
        function draw() {
            let myCanvas = document.getElementById("my-canvas");
            if (myCanvas.getContext) {
                var ctx = myCanvas.getContext('2d');
                makePieChart(ctx, [10, 10, 10, 10, 40, 20]);
            } else {
                alert("Canvas is not supported.");
            }
        }
        function makePieChart(ctx, shares) {
            let sum = shares.reduce((a, b) => a + b, 0);
            if (sum !== 100)
                throw new Error("Sum of all shares must be 100.");
            let startAngle = 0;
            for (let i = 0; i < shares.length; i++) {
                let endAngle = startAngle + (Math.PI / 180) * shares[i] *
3.6;
                ctx.beginPath();
                ctx.arc(200, 150, 120, startAngle, endAngle);
                ctx.lineTo(200, 150);
                ctx.closePath();
                ctx.stroke();
                startAngle = endAngle;
            }
        }
    </script>
</body>
</html>
```

Unutar `draw()` funkcije na mestu na kome smo u ovoj lekciji pisali kod za crtanje, sada se nalazi poziv jedne funkcije, koju smo samostalno kreirali. Reč je o funkciji `makePieChart()`. To je funkcija unutar koje je objedinjena logika za crtanje kružnog dijagrama.

Funkcija `makePieChart()` prihvata dva parametra:

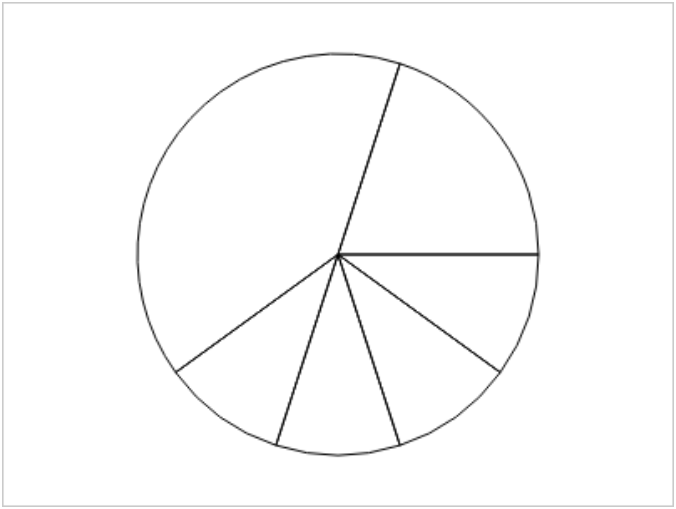
- prvi parametar jeste referenca na kontekst za crtanje
- drugi parametar je niz kojim se funkciji za kreiranje dijagrama prosleđuju udeli koje će imati različiti segmenti dijagrama

Na početak funkcije `makePieChart()` postavljena je provera koja za cilj ima da utvrdi da li je zbir prosleđenih udela jednak 100. Ukoliko nije, podiže se izuzetak.

Kada se funkciji `makePieChart()` proslede udeli čiji zbir iznosi tačno 100, započinje crtanje dijagrama. Crtanje dijagrama se obavlja crtanjem kružnih lukova, čije se ivice spajaju sa centrom kruga, čime se dobijaju pojedinačni isečci kružnog dijagrama.

Crtanje različitih delova dijagrama obavlja se unutar jedne for petlje, pomoću koje se prolazi kroz sve elemente prosleđenog niza. Jedino što se menja u svakoj narednoj iteraciji jeste početni i završni ugao. Početni ugao započinje od 0, a onda u svakoj narednoj iteraciji postaje završni ugao prethodnog segmenta dijagrama.

Kako bi se formirale vrednosti uglova u radijanima, u primeru se sprovode dve konverzije. Prvo se prosleđeni procenti množe se 3.6 kako bi se dobila vrednost u stepenima. Tako dobijeni stepeni se u radijane pretvaraju korišćenjem formule `(Math.PI/180)*degrees`. Sve ovo na kraju rezultuje dijagramom kao na slici 5.17.



Slika 5.17 - Dinamički kreiran kružni dijagram

Kompletan kod primera možete da preuzmete sa sledećeg linka:

```
canvas-example-piechart.rar
```

Rezime

- crtanje bilo koje grafike unutar `canvas`-a, podrazumeva poznavanje osobina koordinatnog sistema `canvas` elementa
- referenta tačka koordinatnog sistema `canvas` elementa nalazi se u gornjem, levom uglu; reč je o početnoj tački koordinatnog sistema, koja ima koordinate (0, 0)
- Canvas API poseduje dva ugrađena mehanizma za crtanje geometrijskih oblika: crtanje pravougaonika i crtanje putanja
- metoda `fillRect()` koristi se za crtanje pravougaonika sa ispunom
- metoda `strokeRect()` koristi se za crtanje pravougaonika bez ispune, odnosno samo sa okvirima
- metoda `clearRect()` koristi se za brisanje pravougaonog prostora unutar `canvas`-a
- putanje su sačinjene iz tačaka koje su međusobno povezane linijama, pri čemu takve linije mogu biti prave, ali i krive
- korišćenjem putanja moguće nacrtati bilo koji geometrijski oblik
- crtanje putanje započinje pozivanjem metode `beginPath()`
- imaginarnu olovku za crtanje je moguće pomerati korišćenjem metode `moveTo()`
- osobine putanje je moguće definisati korišćenjem brojnih metoda, kao što su `lineTo()`, `arc()`, `rect()`…
- konkretno crtanje putanje se obavlja jednom od dve metode: `stroke()` ili `fill()`
- metodom `stroke()` crtaju se samo okviri oblika koji se dobija kreiranom putanjom
- metodom `fill()` crtaju se oblici sa ispunom koji su ograničeni jednom putanjom
- automatsko zatvaranje putanje obavlja se isključivo od strane metode `fill()`, ali ne i od strane metode `stroke()`
- putanja se eksplicitno može zatvoriti korišćenjem metode `closePath()`