CreateJS – crtanje, stilizovanje i obrada korisničke interakcije

U lekciji pred vama po prvi put uplovljavamo u svet crtanja i animacije korišćenjem neke eksterne biblioteke. Tako će lekcije ovoga modula biti posvećene jednom skupu biblioteka koji se naziva CreateJS. Imaćete prilike da vidite u kojoj meri takve biblioteke mogu da olakšaju posao koji smo mi do sada obavljali korišćenjem izvornog skupa funkcionalnosti objedinjenih unutar Canvas API-ja. Ipak, CreateJS je mnogo više od biblioteke za crtanje, pa tako poseduje funkcionalnosti koje olakšavaju i mnoge druge aspekte kreiranja bogatog multimedijalnog sadržaja unutar HTML dokumenata.

Šta je CreateJS?

CreateJS je skup JavaScript biblioteka koje omogućavaju lako kreiranje interaktivnog sadržaja unutar web pregledača korišćenjem osnovnog skupa web tehnologija.

Slika 12.1. CreateJS (izvor https://www.createjs.com/)

CreateJS je sačinjen iz sledećih potpuno nezavisnih JavaScript biblioteka:

- **EaselJS** – olakšava rad sa canvas elementom,
- **TweenJS** – omogućava lako kreiranje animacija uticanjem na vrednosti HTML atributa i JavaScript svojstava,
- **SoundJS** – olakšava rad sa zvukom,
- **PreloadJS** – omogućava lak rad sa resursima kao što su slike, vektori i zvučni efekti.

Svaka biblioteka koja je sastavni deo CreateJS-a u potpunosti je nezavisna, što znači da je njih moguće koristiti i kombinovati po sopstvenom nahođenju. Mi ćemo se u nastavku ovoga modula upoznati sa svakom od upravo navedenih biblioteka.

EaselJS

EaselJS biblioteka omogućava lako crtanje interaktivnog 2D sadržaja unutar canvas elementa. To praktično znači da EaselJS poseduje funkcionalnosti koje olakšavaju crtanje i stilizovanje geometrijskih oblika, slika i teksta. Pri tome, EaselJS čuva referencu na sve nacrtane oblike, pa tako omogućava da se na veoma lak način obradi korisnička interakcija nad grafikom koja se crta unutar canvas elementa.

Slika 12.2. EaselJS (izvor https://www.createjs.com/)

Copyright © Link group

EaselJS je odličan izbor ukoliko je potrebno napraviti bilo koju vrstu interaktivnog grafičkog sadržaja kao što su dijagrami, grafikoni, mape, galerije, video-igre... Reč je o biblioteci koja ne poseduje zavisnosti, što praktično znači da ne zahteva nijednu drugu biblioteku za svoje funkcionisanje. Takođe, EaselJS je moguće koristiti samostalno ili kombinovati sa bilo kojom bibliotekom iz CreateJS skupa.

Integracija EaselJS biblioteke

Preduslov za korišćenje EaselJS biblioteke jeste njena integracija unutar HTML dokumenta u kome će biti korišćena. Integracija podrazumeva uključivanje jednog JavaScript fajla. Najbolja opcija je uključivanje fajla sa CreateJS CDN servera:

```
<script src="https://code.createjs.com/1.0.0/easeljs.min.js"></script>
```

Objedinjeno uključivanje svih CreateJS biblioteka

EaselJS biblioteku je u tekući HTML dokument moguće uključiti i zajedno sa svim ostalim bibliotekama CreateJS skupa. Stoga, ukoliko planirate da koristite sve CreateJS biblioteke, njihovo objedinjeno uključivanje možete obaviti korišćenjem samo jednog `.js` fajla na sledeći način:

```
<script
src="https://code.createjs.com/1.0.0/createjs.min.js"></script>
```

S obzirom na to da je EaselJS namenjen crtanju grafike unutar `canvas` elementa, dokumentu je neophodno dodati i `canvas` element korišćenjem pristupa koji su ilustrovani u prethodnim modulima ovoga kursa:

```
<canvas id="my-canvas" width="400" height="300">
Your web browser does not support canvas element.
</canvas>
```

Kao i prilikom direktnog rada sa `canvas` elementom, neophodno je da sačekamo da web pregledač u potpunosti učita kompletan HTML dokument. Stoga je potrebno obaviti pretplatu na `load` događaj `window` objekta:

```
window.onload = init;

function init() {

}
```

Kompletan HTML dokument sa uključenom EaselJS bibliotekom, `canvas` elementom za crtanje i pretplatom na `load` događaj izgleda ovako:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>CreateJS Drawing</title>
```

Copyright © Link group

```
    <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
src="https://code.createjs.com/1.0.0/easeljs.min.js"></script>
    <script>
        window.onload = init;

        function init() {

        }
    </script>
</body>

</html>
```

Baš kao i prilikom direktnog rada sa Canvas API-jem, i sada je na `canvas` element postavljen okvir od jednog piksela kako biste lakše mogli da percipirate granice `canvas` elementa unutar HTML dokumenta.

Kod koji se bude prikazivao u nastavku lekcije potrebno je pisati unutar `init()` funkcije.

Pitanje

EaselJS biblioteka je namenjena:

a) crtanju grafike unutar canvas elementa
b) crtanju grafike unutar svg elementa
c) radu sa zvučnim efektima
d) učitavanju resursa

Objašnjenje

EaselJS biblioteka omogućava lako crtanje interaktivnog 2D sadržaja unutar canvas elementa.

Kreiranje pozornice (Stage)

EaselJS biblioteka poznaje pojam pozornice (engl. *stage*), što je koreni element koji objedinjuje svu grafiku koja se crta unutar `canvas` elementa. Reč je o apstrakciji koju EaselJS koristi kako bi mogao da čuva referencu na sve elemente koji se crtaju unutar `canvasa`. Ovo na kraju znači da se kompletna interakcija sa `canvas` elementom korišćenjem EaselJS biblioteke obavlja posredstvom pozornice. Može se reći da je pozornica naš kontekst za crtanje kada se koristi EaselJS biblioteka.

Copyright © Link group

Kako bi se kreirala pozornica za crtanje, neophodno je doći do reference na canvas element i takvu referencu proslediti Stage klasi EaselJS biblioteke prilikom instanciranja:

```
let canvas = document.getElementById("my-canvas");
let stage = new createjs.Stage(canvas);
```

Prvom prikazanom naredbom dolazi se do DOM objekta, koji predstavlja canvas element. Takva referenca prosleđuje se konstruktoru Stage klase. Tako se drugom naredbom obavlja kreiranje jednog objekta koji predstavlja pozornicu za crtanje.

Prostor imena createjs

Prva značajna osobina EaselJS biblioteke, koju možemo videti iz prikazanih naredbi, jeste upotreba prostora imena createjs. Naime, sve funkcionalnosti EaselJS biblioteke grupisane su unutar nekoliko klasa. Sve takve klase se nalaze unutar prostora imena createjs, čime se sprečava eventualni konflikt imena do koga može doći kada se u jednom projektu koristi veći broj različitih biblioteka.

Pored prostora imena createjs, upravo prikazane naredbe za kreiranje pozornice ilustruju korišćenje još jednog značajnog elementa EaselJS biblioteke – klase Stage.

Klasa Stage

Stage je osnovna klasa EaselJS biblioteke, kojom se objedinjuje sva grafika koja se crta unutar canvas elementa. Konstruktor ove klase prihvata jedan parametar:

```
Stage (canvas)
```

Konstruktoru se može proslediti DOM objekat koji predstavlja canvas element ili string, koji predstavlja vrednost id atributa canvas elementa. Tako se upravo prikazane dve naredbe mogu pretvoriti u jednu:

```
let stage = new createjs.Stage("my-canvas");
```

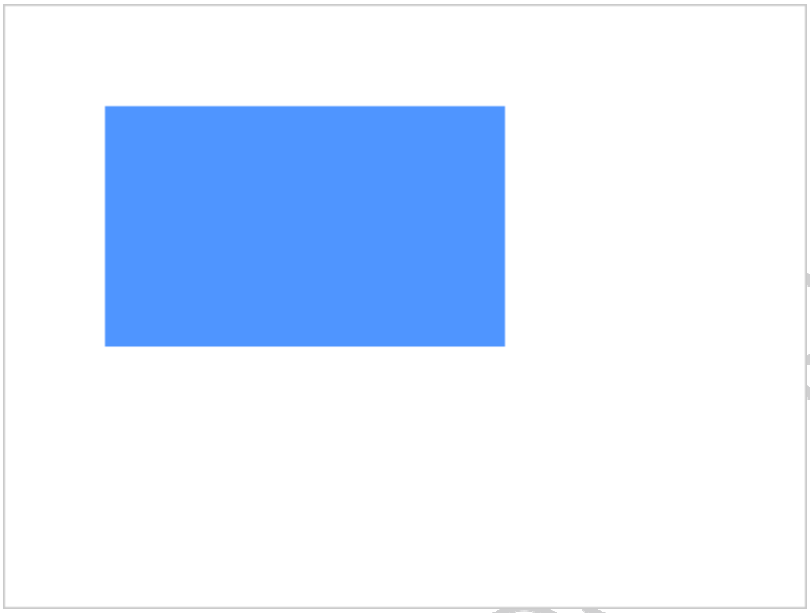
Crtanje prve grafike korišćenjem EaselJS-a

Nakon uspostavljanja okruženja i dolaska do pozornice, odmah ćemo preći na crtanje prve grafike unutar canvasa korišćenjem EaselJS biblioteke. To će nam pomoći da se upoznamo sa osnovnim elementima koji učestvuju u takvom procesu. Biće nacrtan jedan pravougaonik sa ispunom, a kod kojim će to biti realizovano izgleda ovako:

```
let rect = new createjs.Shape();
rect.graphics.beginFill("#4F95FF").drawRect(50, 50, 200, 120);
stage.addChild(rect);

stage.update();
```

Copyright © Link group

Kada nakon ovih izmena pogledate unutar canvas elementa, moći ćete da vidite efekat kao na slici 12.3.



Slika 12.3. Prva grafika nacrtana korišćenjem EaselJS biblioteke

Crtanje pravougaonika koji možete videti na slici 12.3. obavljeno je korišćenjem nekoliko naredbi. Prvom naredbom je obavljeno instanciranje `Shape` klase, te dobijanje objekta tipa `Shape`. Zatim su nad `graphics` svojstvom takvog objekta definisane osobine grafike koja će biti nacrtana. Svojstvo `graphics` čuva referencu na objekat tipa `Graphics`. Tako su `Shape` i `Graphics` dve osnove klase koje se koriste prilikom crtanja unutar pozornice EaselJS biblioteke.

Klase Shape i Graphics

Korišćenjem klase `Shape` kreiraju se objekti koji predstavljaju oblike koji će biti nacrtani unutar pozornice EaselJS biblioteke. Objekat `Shape` klase moguće je kreirati korišćenjem podrazumevanog konstruktora bez parametara, baš kao što je to učinjeno u primeru. Ipak, to je moguće obaviti i prosleđivanjem objekta tipa `Graphics`:

```
Shape (graphics)
```

Iz ovoga se može videti da se u procesu crtanja grafike unutar pozornice EaselJS biblioteke klase `Shape` i `Graphics` moraju posmatrati u zajedničkom kontekstu. `Shape` omogućava da se oblik prikaže unutar pozornice, a `Graphics` obezbeđuje metode za definisanje osobina grafike.

Instancu klase `Graphics` `Shape` objekat može da dobije na dva načina:

- kroz konstruktor,
- definisanjem vrednosti svojstva `graphics`.

Copyright © Link group

U prikazanom primeru je instanca Graphics klase definisana korišćenjem istoimenog svojstva. Identično je moglo biti postignuto i prethodnim kreiranjem objekta tipa Graphics, a zatim njegovim prosleđivanjem konstruktoru klase Shape:

```
let g = new createjs.Graphics();
g.beginFill("#4F95FF");
g.drawRect(50, 50, 200, 120);
let rect = new createjs.Shape(g);
```

Sada je obavljeno identično, ali je prvo konstruisan objekat tipa Graphics, koji je zatim prosleđen konstruktoru klase Shape.

Između dva prikazana pristupa za rad sa Shape i Graphics objektima možete da uvidite još jednu razliku. U prvom primeru su pozivi svih metoda Graphics klase nadovezani, dok su u drugom primeru oni izolovani u zasebne naredbe. Jednostavno, sve metode Graphics klase kao svoju povratnu vrednost emituju tekuću Graphics instancu, što omogućava njihovo nadovezivanje.

U prethodnim redovima mogli ste da pročitate da Shape omogućava da se oblik prikaže unutar canvas elementa, a da Graphics obezbeđuje metode za definisanje osobina grafike. Ipak, kako bi grafika zaista postala vidljiva unutar canvas elementa, neophodno je obaviti još dve operacije:

- dodavanje oblika pozornici,
- ažuriranje pozornice.

Metoda addChild()

Ukoliko želimo da se neki oblik pojavi unutar canvas elementa, neophodno ga je dodati pozornici. Dodavanje grafike unutar pozornice obavlja se korišćenjem metode addChild(), klase Stage:

```
addChild(childs...)
```

Metoda addChild() može da prihvati proizvoljan broj parametara. U prikazanom primeru njoj je prosleđen jedan parametar:

```
stage.addChild(rect);
```

Ipak, kada pozornici želimo da dodamo veći broj oblika odjednom, moguće je napisati i nešto ovako:

```
stage.addChild(rect1, rect2, circle1, circle2);
```

Korišćenjem metode addChild() obavlja se dodavanje nekog oblika, odnosno grafike, navrh liste elemenata koji se prikazuju unutar canvasa. To znači da će poslednje prosleđen element unutar canvasa biti prikazan iznad ostalih elemenata.

Copyright © Link group

Metoda update()

Dodavanje nekog elementa unutar pozornice nije dovoljno da bi takav element postao vidljiv unutar `canvas` elementa. Neophodno je obaviti još jednu operaciju koja se ogleda u pozivanju metode `update()`. Pozivanjem ove metode inicira se crtanje svih objekata koji su prethodno dodati pozornici.

Crtanje i stilizovanje geometrijskih oblika

U prethodnim redovima mogli ste da vidite da su dva osnova činioca u procesu crtanja geometrijskih oblika unutar pozornice EaselJS biblioteke klase `Shape` i `Graphics`. Stoga ćemo se u nastavku malo detaljnije upoznati sa skupom funkcionalnosti koje ove klase poseduju. To će nam na kraju omogućiti da obavimo crtanje i stilizovanje različitih geometrijskih oblika.

Klasa `Graphics` unutar sebe objedinjuje funkcionalnosti za crtanje i stilizovanje grafike. Stoga se objekti ove klase mogu uporediti sa objektom konteksta za crtanje, nad kojim smo u prethodnim modulima definisali osobine grafike koju smo crtali, prilikom direktnog korišćenja Canvas API-ja. Ipak, klasa `Graphics` unutar sebe poseduje i brojne funkcionalnosti koje nisu deo izvornog skupa funkcionalnosti Canvas API-ja. Najznačajnije metode klase `Graphics`, kada je u pitanju crtanje geometrijskih oblika, ilustrovane su tabelom 12.1.

Metoda	Opis
`moveTo (x, y)`	Pomera imaginarnu olovku za crtanje na novu poziciju definisanu prosleđenim koordinatama
`lineTo (x, y)`	Crta liniju spajanjem trenutne tačke i tačke čije se koordinate prosleđuju kao parametri
`arc ( x, y, radius, startAngle, endAngle, anticlockwise )`	Crta kružni luk, baš kao i istoimena metoda Canvas API-ja

`drawRect (x, y, width, height)`
 	
`rect (x, y, width, height)`
	Crtaju pravougaonik, baš kao i slične metode Canvas API-ja

`drawRoundRect (x, y, width, height, radius)`	Crta pravougaonik sa zaobljenim ivicama; sve ivice imaju identično zaobljenje koje se definiše korišćenjem parametra `radius`
`drawRoundRectComplex (x, y, w, h, radiusTL, radiusTR, radiusBR, radiusBL)`	Crta pravougaonik sa zaobljenim ivicama, pri čemu je moguće kontrolisati zaobljenje svake pojedinačne ivice korišćenjem zasebnih parametara
`drawCircle (x, y, radius)`	Crta krug
`drawEllipse (x, y, width, height)`	Crta elipsu

Tabela 12.1. Metode za crtanje klase Graphics

Copyright © Link group

Metode klase `Graphics` za stilizovanje geometrijskih oblika ilustrovane su tabelom 12.2.

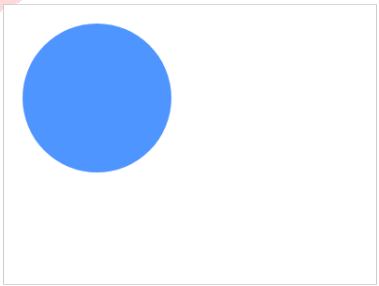
Metoda	Opis
`beginFill (color)`	Definiše boju ispune
`beginStroke (color)`	Definiše boju okvira
`setStrokeStyle (thickness, [caps=0], [joints=0], [miterLimit=10], [ignoreScale=false])`	Definiše stilizaciju linija
`setStrokeDash ([segments], [offset=0])`	Omogućava definisanje isprekidanih linija
`beginLinearGradientFill (colors, ratios, x0, y0, x1, y1)`	Definiše linearni gradijent koji će biti iskorišćen za postavljanje boje ispune
`beginLinearGradientStroke (colors, ratios, x0, y0, x1, y1)`	Definiše linearni gradijent koji će biti iskorišćen za postavljanje boje okvira
`beginRadialGradientFill (colors, ratios, x0, y0, r0, x1, y1, r1 )`	Definiše radijalni gradijent koji će biti iskorišćen za postavljanje boje ispune
`beginRadialGradientStroke (colors, ratios, x0, y0, r0, x1, y1, r1)`	Definiše radijalni gradijent koji će biti iskorišćen za postavljanje boje okvira

Tabela 12.2. Metode za stilizovanje klase Graphics

Unutar tabela 12.1. i 12.2. možete videti da EaselJS poseduje najznačajnije metode za crtanje i stilizaciju sa kojima smo se već upoznali prilikom rada sa Canvas API-jem, ali i veliki broj dodatnih metoda koje izvorni skup funkcionalnosti ne poseduje. Na primer, EaselJS poseduje metodu koja omogućava direktno crtanje kruga:

```
let g = new createjs.Graphics();
g.beginFill("#4F95FF");
g.drawCircle(100, 100, 80);
let circle = new createjs.Shape(g);
stage.addChild(circle);
```

Na ovaj način se unutar canvas elementa dobija grafika kao na slici 12.4.



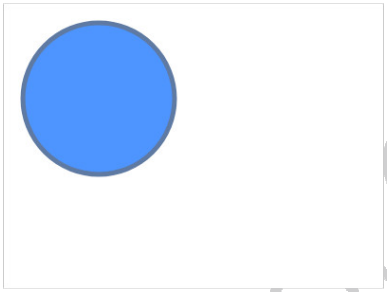
Slika 12.4. Krug nacrtan korišćenjem EaselJS-a

Copyright © Link group

Još jedna od značajnih prednosti crtanja korišćenjem EaselJS-a ogleda se u mogućnosti definisanja oblika koji poseduju i ispunu i okvire:

```
let g = new createjs.Graphics();
g.beginFill("#4F95FF");
g.beginStroke("#5F7AA3");
g.setStrokeStyle(5);
g.drawCircle(100, 100, 80);
let circle = new createjs.Shape(g);
stage.addChild(circle);
```

Efekat je kao na slici 12.5.



Slika 12.5. Krug sa ispunom i okvirima

Identičan efekat se može postići i korišćenjem kompaktnije sintakse, koju ste mogli da vidite nešto ranije, a koja podrazumeva nadovezivanje metoda:

```
let circle = new createjs.Shape();
circle.graphics.beginFill("#4F95FF").beginStroke("#5F7AA3").setStrokeSt
yle(5).drawCircle(100, 100, 80);
stage.addChild(circle);
```

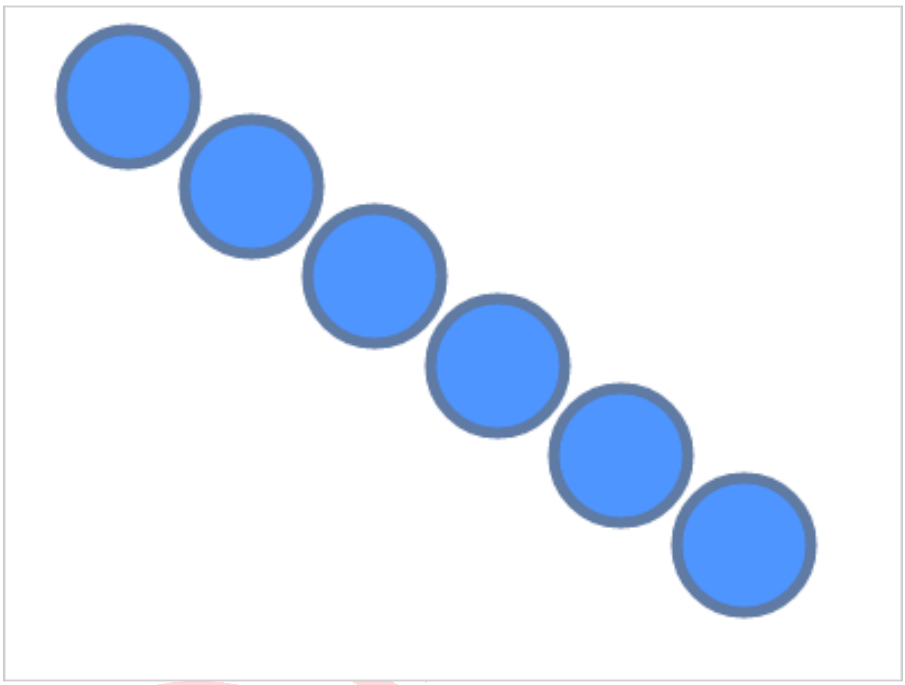
Mi ćemo u nastavku isključivo koristiti ovu kompaktniju sintaksu.

Postojanje klasa `Shape` i `Graphics`, kojima se razdvajaju operacije definisanja osobina grafike od njenog prikaza, donosi još jednu prednost EaselJS sistema za crtanje. Naime, osobine nekog oblika moguće je definisati jednom, a zatim takav oblik prikazivati proizvoljan broj puta, na različitim pozicijama unutar pozornice:

```
let g = new createjs.Graphics();
g.beginFill("#4F95FF").beginStroke("#5F7AA3").setStrokeStyle(5).drawCir
cle(0, 0, 30);

for (let i = 1; i < 7; i++) {
    let circle = new createjs.Shape(g);
    circle.x = i * 55;
    circle.y = i * 40;
    stage.addChild(circle);
}
```

Copyright © Link group

Ovo je primer koji podrazumeva kreiranje jednog objekta tipa `Graphics` koji se koristi unutar nekoliko objekata tipa `Shape`. Objekat tipa `Graphics` poseduje osobine kojima se dobija već viđeni krug sa ispunom i okvirima. Ipak, ovoga puta su koordinate centra kruga prilikom konstruisanja `Graphics` objekta postavljene na (0,0). Unutar for petlje obavlja se kreiranje šest objekata tipa `Shape` na osnovu identičnog, prethodno kreiranog `Graphics` objekta. Tom prilikom se obavlja i dinamičko definisanje koordinata centra svakog od kruga. Jednostavno, objekti tipa `Shape` poseduju svojstva `x` i `y`, kojima je moguće definisati poziciju na kojoj će se oblik naći unutar pozornice. Na ovaj način se unutar `canvasa` dobija efekat kao na slici 12.6.



Slika 12.6. Šest krugova nacrtanih na osnovu istog Graphics objekta

Obrada korisničke interakcije

Najveća prednost koju donosi korišćenje EaselJS sistema za crtanje jeste svakako mogućnost rukovanja elementima koji se crtaju kao da je reč o elementima HTML dokumenta. Naime, EaselJS čuva referencu na svaki nacrtani oblik, što nama omogućava da obavimo pretplatu na neki od događaja do kojih može doći nad oblicima koji se crtaju:

```
let circle = new createjs.Shape();
circle.graphics.beginFill("#4F95FF").beginStroke("#5F7AA3").setStrokeSt
yle(5).drawCircle(100, 100, 80);
stage.addChild(circle);


circle.addEventListener("click", function() {
    alert("You have clicked on circle!");
});
```

Copyright © Link group

Možete videti da je prikazanim kodom obavljena pretplata na `click` događaj, nad objektom tipa `Shape`. Kada se klikne na nacrtani krug, biće izvršena logika funkcije koja je kao drugi parametar prosleđena metodi `addEventListener()`. U našem slučaju to će podrazumevati prikaz modalnog prozora sa porukom *You have clicked on circle!*

Da bi se ovako nešto obavilo korišćenjem Canvas API-ja, bilo bi neophodno detektovati koordinate tačke na kojoj se pokazivač miša nalazio tokom klika. Zatim bi bilo potrebno izvršiti proveru da li su koordinate takve tačke u okvirima nacrtanog kruga. Sve to za nas sada obavlja EaselJS biblioteka.

EaselJS omogućava pretplatu na nekoliko događaja prikazanih tabelom 12.3.

Događaj	Opis
`click`	Klik – podrazumeva pritisak tastera miša i njegovo otpuštanje i sve to dok je pokazivač miša iznad elementa čiji se događaj sluša
`mousedown`	Događaj koji se aktivira čim korisnik pritisne taster miša; drugim rečima, emitovanje događaja ne čeka da korisnik otpusti taster miša kao kod događaja `click`
`dblclick`	Događaj koji se aktivira prilikom duplog klika
`pressmove`	Događaj koji se aktivira kada korisnik pritisne taster miša, a onda, držeći taster pritisnut, pomera kursor miša
`pressup`	Događaj koji se aktivira kada se taster miša otpusti dok je pokazivač iznad elementa nad kojim je definisana pretplata
`mouseover`	Događaj koji se aktivira kada se pokazivač miša nađe iznad nekog elementa
`mouseout`	Događaj koji se aktivira kada pokazivač miša izađe iz okvira nekog elementa
`rollover`	Događaj koji se aktivira kada se pokazivač miša nađe iznad nekog elementa; za razliku od `mouseover` događaja, `rollover` se aktivira samo jednom, bez obzira na to iznad koliko elemenata naslednika se pokazivač miša nalazi
`rollout`	Događaj koji je gotovo identičan `mouseout` događaju, uz razliku koja se odnosi na tretiranje elemenata naslednika

Tabela 12.3. Događaji miša koje je moguće slušati

Copyright © Link group

Napomena

Kako bi bilo moguće slušati `mouseover/mouseout` i `rollover/rollout` događaje, neophodno je pozvati sledeću metodu:

```
stage.enableMouseOver(10);
```

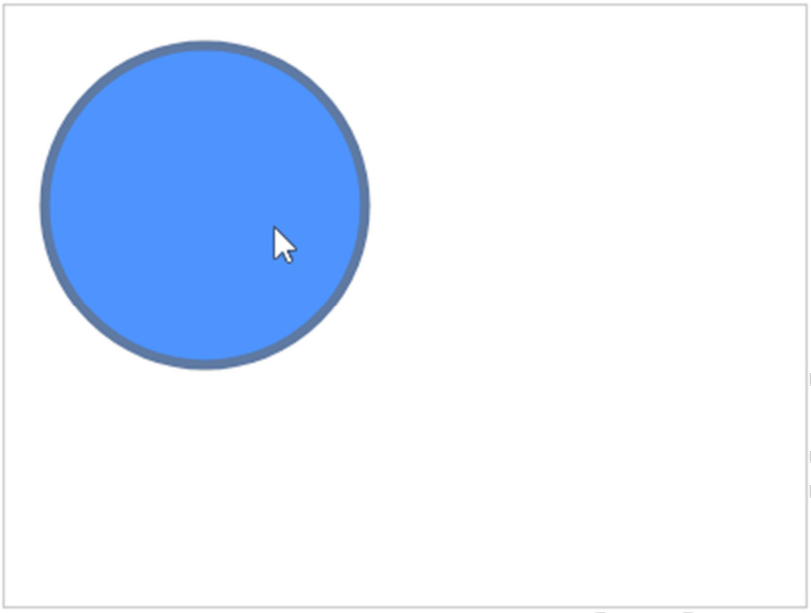
U prikazanoj naredbi se poziva `enableMouseOver()` metoda `Stage` klase, kojom se aktiviraju četiri spomenuta događaja. Naime, navedeni događaji su najzahtevniji za slušanje, s obzirom na to da zahtevaju da EaselJS konstantno prati koordinate pokazivača miša. Pošto ne postoji izvorni mehanizam za obavljanje takvog posla, EaselJS spomenute događaje sluša periodičnom proverom koordinata na kojima se nalazi pokazivač miša. Parametar koji se prosleđuje metodi `enableMouseOver()` odnosi se na period između dve takve provere. Podrazumevani period osvežavanja je 20 ms.

Evo nešto korisnijeg primera obrade `click` događaja koji će ilustrovati još jednu zanimljivu mogućnost EaselJS biblioteke:

```
let stage = new createjs.Stage("my-canvas");

let circle = new createjs.Shape();

let fillCommand = circle.graphics.beginFill("#4F95FF").command;

circle.graphics.beginStroke("#5F7AA3").setStrokeStyle(5).drawCircle(100
, 100, 80);
stage.addChild(circle);
stage.enableMouseOver();

circle.addEventListener("click", function(evt) {

    let red = Math.floor(Math.random() * 256);
    let green = Math.floor(Math.random() * 256);
    let blue = Math.floor(Math.random() * 256);

    fillCommand.style = `rgb(${red}, ${green}, ${blue})`;
    stage.update();
});


stage.update();
```

Efekat prikazanog koda je kao na animaciji 12.1.

Copyright © Link group



Animacija 12.1. Promena boje pozadine nacrtanog oblika na klik

Prikazani primer ilustruje promenu boje pozadine već nacrtanog kruga prilikom klika na njega. Za realizaciju primera iskorišćen je dobar deo već viđene logike, uz dodatak jedne funkcionalnosti koju do sada nismo videli. Reč je o komandama.

EaselJS komande

Poziv svake metode klase `Graphics` proizvodi novi objekat komande. Takvom objektu je moguće pristupiti korišćenjem svojstva `command`. Upravo to je u primeru učinjeno korišćenjem sledeće naredbe:

```
let fillCommand = circle.graphics.beginFill("#4F95FF").command;
```

Na ovaj način je obavljeno isto ono što i do sada – postavljanje boje ispune oblika koji će biti nacrtan. Ipak, takva operacija je sada uhvaćena u jedan objekat komande čija je referenca smeštena unutar promenljive `fillCommand`.

Ovako nešto nam omogućava da kasnije na veoma lak način obavimo izmenu vrednosti koja se odnosi na boju ispune:

```
fillCommand.style = `rgb(${red}, ${green}, ${blue})`;
```

Korišćenjem svojstva `style` utiče se na boju pozadine već nacrtanog kruga.

Crtanje teksta

EaselJS omogućava da se unutar `canvas` elementa obavi crtanje teksta. Procedura crtanja teksta podrazumeva prethodno kreiranje objekta tipa `Text`.

Copyright © Link group

Klasa Text

Crtanje teksta obavlja se korišćenjem klase `Text`. Ova klasa poseduje konstruktor sledećih osobina:

```
Text ([text], [font], [color])
```

Možete videti da su svi parametri opcioni, što praktično znači je klasu `Text` moguće instancirati i korišćenjem podrazumevanog konstruktora bez parametara, a zatim vršiti postavljanje vrednosti svojstava nad dobijenim objektom.

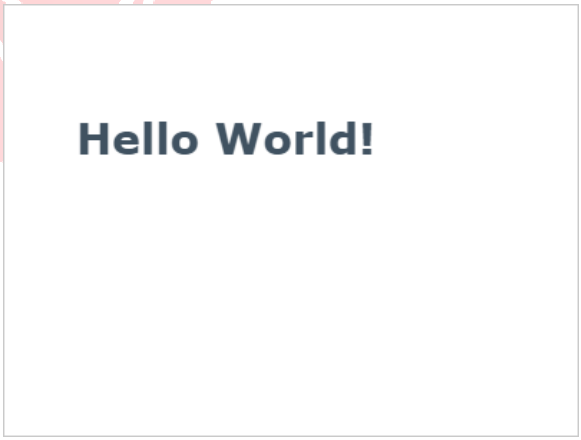
Primer crtanja teksta može da izgleda ovako:

```
let stage = new createjs.Stage("my-canvas");

let text = new createjs.Text("Hello World!", "bold 30px Verdana",
"#405261");
text.x = 50;
text.y = 80;

stage.addChild(text);

stage.update();
```

U primeru je prvo obavljeno instanciranje klase `Text` i tom prilikom su konstruktoru prosleđena 3 parametra:

- tekst koji je potrebno ispisati,
- stil teksta, odnosno stil, veličina i familija fonta,
- boja teksta.

Unutar `canvas` elementa dobija se efekat kao na slici 12.7.



Slika 12.7. Tekst koji je nacrtan unutar canvasa

Copyright © Link group

Veličinu teksta unutar canvas elementa moguće je dobiti korišćenjem metode getBounds().

Metoda getBounds()

Za dobijanje osobina imaginarnih okvira u kojima se nalazi tekst koji se crta unutar pozornice EaselJS sistema moguće je koristiti metodu getBounds(). Ova metoda može se upotrebiti na sledeći način:

```
let bounds = obj.getBounds();
```

Povratna vrednost metode getBounds() jeste objekat tipa Rectangle, koji poseduje sledeća svojstva:

- x – koordinata referentne tačke po x osi,
- y – koordinata referentne tačke po y osi,
- width – širina nacrtane grafike,
- height – visina nacrtane grafike.

Primer: Crtanje Pie Chart dijagrama

Neki od pristupa sa kojima smo se upoznali u ovoj lekciji sada će biti iskorišćeni za crtanje *Pie Chart* dijagrama. Biće to dijagram po ugledu na onaj koji smo već videli u prethodnim modulima ovoga kursa.

Kod primera izgledaće ovako:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>CreateJS - Pie Chart Diagram</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
src="https://code.createjs.com/1.0.0/easeljs.min.js"></script>
    <script>
        window.onload = init;
        let stage;
```

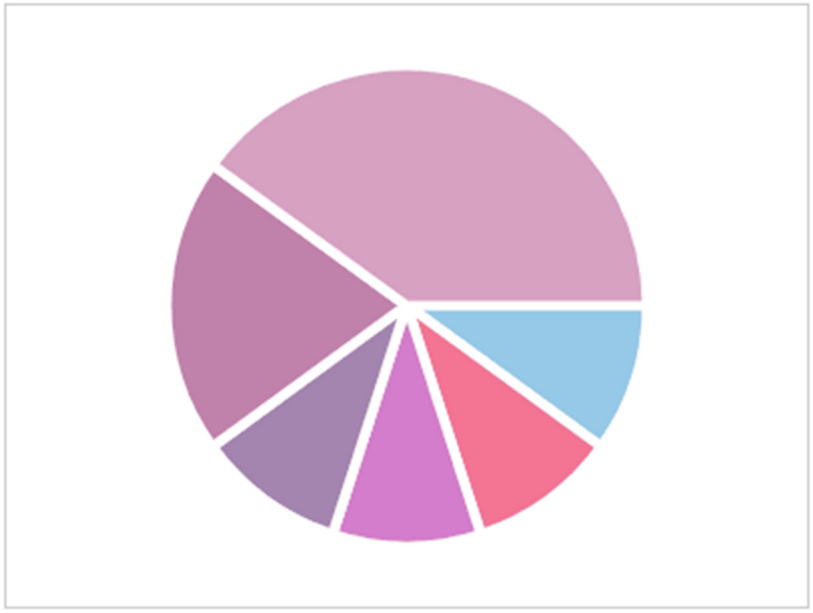
Copyright © Link group

```
            function init() {
                stage = new createjs.Stage("my-canvas");
                stage.enableMouseOver(10);

                makePieChart(stage, [10, 10, 10, 10, 20, 40]);

                stage.update();
            }

            let fillCommand;

            function makePieChart(stage, 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;

                    let segment = new createjs.Shape();

                    let red = Math.floor(Math.random() * 200) + 50;
                    let green = Math.floor(Math.random() * 200) + 50;
                    let blue = Math.floor(Math.random() * 200) + 50;

                    segment.fillCommand =
    segment.graphics.beginFill('rgba(' + red + ', ' + green + ', ' + blue
    + ', 0.7 )').command;


    segment.graphics.beginStroke('white').setStrokeStyle(5);

                    segment.graphics.arc(200, 150, 120, startAngle,
    endAngle).lineTo(200, 150).closePath();

                    stage.addChild(segment);

                    startAngle = endAngle;

                    segment.addEventListener("mouseover", onMouseOver);
                    segment.addEventListener("mouseout", onMouseOut);


                }
            }

            function onMouseOver(event) {
                event.target.fillCommand.style =
    event.target.fillCommand.style.replace('0.7', '1.0');
                stage.update();
            }
```

Copyright © Link group

```
            function onMouseOut(event) {
                event.target.fillCommand.style =
event.target.fillCommand.style.replace('1.0', '0.7');
                stage.update();
            }
        </script>
    </body>

    </html>
```

Efekat koda je kao na animaciji 12.2.



Animacija 12.2. Pie Chart dijagram nacrtan korišćenjem EaselJS biblioteke

Kompletan kod možete da preuzmete sa sledeće adrese:

`createjs_piechart_diagram.rar`

Rezime

- CreateJS je skup JavaScript biblioteka koje omogućavaju lako kreiranje interaktivnog sadržaja unutar web pregledača.
- CreateJS je sačinjen iz sledećih biblioteka: EaselJS, TweenJS, SoundJS i PreloadJS.
- EaselJS biblioteka omogućava lako crtanje interaktivnog 2D sadržaja unutar `canvas` elementa.
- Preduslov za korišćenje EaselJS biblioteke jeste njena integracija unutar HTML dokumenta uključivanjem jednog `.js` fajla.
- EaselJS biblioteka poznaje pojam pozornice (engl. *stage*), što je koreni element koji objedinjuje svu grafiku koja se crta unutar `canvas` elementa.

Copyright © Link group

- Sve funkcionalnosti EaselJS biblioteke grupisane su unutar nekoliko klasa, a sve takve klase nalaze se unutar prostora imena `createjs`.
- `Stage` je osnovna klasa EaselJS biblioteke, kojom se objedinjuje sva grafika koja se crta unutar `canvas` elementa.
- Korišćenjem klase `Shape` kreiraju se objekti koji predstavljaju oblike koji će biti nacrtani unutar pozornice EaselJS biblioteke.
- Klasa `Graphic` obezbeđuje metode za definisanje osobina grafike.
- Dodavanje grafike unutar pozornice obavlja se korišćenjem metode `addChild()`, klase `Stage`.
- Pozivanjem `update()` metode, klase `Stage`, inicira se crtanje svih objekata koji su prethodno dodati pozornici.
- Objekti tipa `Shape` poseduju svojstva `x` i `y`, kojima je moguće definisati poziciju na kojoj će se oblik naći unutar pozornice.
- EaselJS čuva referencu na nacrtane oblike, pa tako omogućava slušanje različitih događaja miša nad njima.
- Pozivanjem metode `enableMouseOver()` omogućava se slušanje događaja `mouseover/mouseout` i `rollover/rollout`.
- Crtanje teksta obavlja se korišćenjem klase `Text`.

Copyright © Link group