Interfejs Canvas

10.1 Grafika za veb

Na početku knjige pomenuli smo da HTML5 zamenjuje ranije dodatne programe, kao što su Flash ili Java apleti. Da bi veb postao nezavisan od tehnologija drugih proizvođača, trebalo je razmotriti bar dva važna aspekta: obradu video-sadržaja i grafičke aplikacije. Video-sadržaji se uspešno obrađuju pomoću elementa **<video>** i API-ja analiziranih u prethodnim poglavljima, ali oni nisu namenjeni za grafiku. HTML5 je uveo interfejs Canvas, koji veoma efikasno radi s grafikom – možete da crtate, prikazujete grafiku, animirate i obrađujete slike i tekst. Zajedno sa ostalim API-jima, Canvas učestvuje u izradi aplikacija, čak i 2D i 3D video-igrica za veb.

Element <canvas>

Element **<canvas>** generiše na veb strani prazan pravougaoni prostor, u kome se prikazuju rezultati metoda API-ja Canvas. Prostor je bela površina, slična onoj koju daje prazan element , ali potpuno drugačije namene.

Listing 10-1: *Upotreba elementa* <canvas>.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Canvas API</title>
  <script src="canvas.js"></script>
</head>
<body>
  <section id="canvasbox">
    <canvas id="canvas" width="500" height="300"></canvas>
  </section>
</body>
</html>
```

Za ovaj element potrebno nam je samo nekoliko atributa, što se vidi iz listinga 10-1. Atributi **width** i **height** deklarišu veličinu okvira. Oni su neophodni, jer će svaki sadržaj prikazan pomoću ovog elementa imati te dimenzije.

Namena elementa **<canvas>** u osnovi je izrada praznog okvira na ekranu. Samo kroz JavaScript, kao i nove metode i svojstva koje uvodi API Canvas, ova površina postaje funkcionalna i može se upotrebiti u praktične svrhe.

> **VAŽNO** Ukoliko API Canvas nije dostupan u čitaču veba, sadržaj naveden između oznaka **<canvas>** prikazuje se na ekranu radi kompatibilnosti.

Metoda getContext()

Metoda **getContext()** prva je koju moramo da pozovemo kako bi element **<canvas>** bio spreman za rad. Ona generiše kontekst za crtanje koji se dodeljuje platnu. Zahvaljujući njemu moći ćemo da primenimo ostale mogućnosti API-ja.

Listing 10-2: *Pravljenje konteksta za crtanje na platnu.*

```
function initiate(){
  var elem = document.getElementById('canvas');
  var canvas = elem.getContext('2d');
}
addEventListener("load", initiate);
```

U listingu 10-2, referenca na element **<canvas>** smešta se u promenljivu **elem**, a kontekst se pravi pomoću metode **getContext('2d')**. Ova metoda može imati dve vrednosti: **2d** i **webgl** (za dvodimenzionalno i trodimenzionalno okruženje). U vreme pisanja ove knjige, vrednost **2d** je dostupna u svakom čitaču veba koji je kompatibilan sa HTML5, a vrednost **webgl** samo u čitačima koji primenjuju biblioteku WebGL za generisanje 3D grafike. O biblioteci WebGL govorićemo u narednom poglavlju.

Kontekst za crtanje na 2D platnu biće pomoćna mreža piksela rapoređenih u redove i kolone odozgo nadole i sleva udesno, pri čemu je koordinatni početak (piksel 0, 0) smešten u gornji levi ugao kvadrata.

> **Uradite sami** Kopirajte HTML dokument iz listinga 10-1 u novu praznu datoteku. Napravite i datoteku sa imenom **canvas.js** i kopirajte u nju sve skriptove, počevši od listinga 10-2. Svaki primer u ovom poglavlju nezavisan je i zamenjuje prethodni.

10.2 Crtanje na platnu

Kada pripremimo element **<canvas>** i kontekst platna, možemo da napravimo i obradimo grafiku. Spisak alatki koje interfejs Canvas nudi za ovu namenu veoma je dug i pomoću njih se može napraviti mnogo toga – od jednostavnih oblika i crteža, do teksta, senki ili složenih transformacija. U ovom odeljku razmotrićemo sve pomenute tehnike pojedinačno.

Crtanje pravougaonika

Najčešće programer mora prvo da pripremi sadržaj koji se crta, i tek nakon toga da ga pošalje u kontekst (što ćemo uskoro videti), ali postoji i nekoliko metoda za direktno crtanje na platnu. Te metode su specifične za pravougaone oblike, i jedine mogu da generišu prost geometrijski oblik. (Da bismo dobili druge oblike, moramo da kombinujemo druge metode za crtanje i složene putanje.) Slede raspoložive metode:

`fillRect(`*`x, y, širina, visina`*`)` – Iscrtava pun pravougaonik. Gornji levi ugao smešta se na položaj zadat atributima ***`x`*** i ***`y`***. Atributi ***`širina`*** i ***`visina`*** deklarišu veličinu pravougaonika.

`strokeRect(`*`x, y, širina, visina`*`)` – Slično prethodnoj metodi, ova iscrtava prazan pravougaonik, tj., samo konturu.

`clearRect(`*`x, y, širina, visina`*`)` – Koristi se za oduzimanje piksela od oblasti koju zadaju atributi metode. Ponaša se slično kao pravougaoni brisač.

Listing 10-3: *Crtanje pravougaonika.*

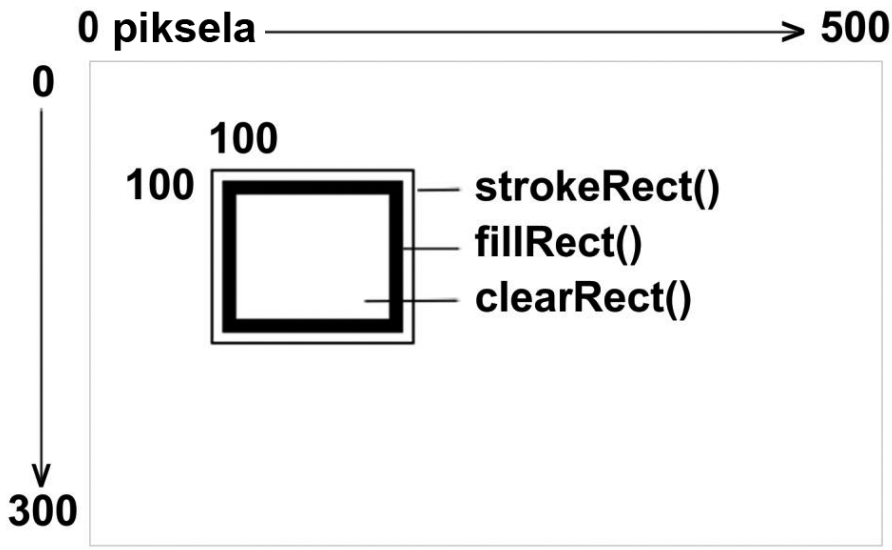
```
function initiate(){
  var elem = document.getElementById('canvas');
  var canvas = elem.getContext('2d');

  canvas.strokeRect(100, 100, 120, 120);
  canvas.fillRect(110, 110, 100, 100);
  canvas.clearRect(120, 120, 80, 80);
}
addEventListener("load", initiate);
```

Ova funkcija je ista kao ona iz listinga 10-2, ali s nekoliko novih metoda koje zaista crtaju na platnu. Kontekst se dodeljuje promenljivoj **`canvas`**, pa se sada ona koristi za referenciranje konteksta u svakoj metodi.

Prva metoda, **`strokeRect(100, 100, 120, 120)`**, crta prazan pravougaonik u gornjem levom uglu na poziciji `100, 100`, i u veličini od 120 piksela. Druga metoda, **`fillRect(110, 110, 100, 100)`**, crta pun pravougaonik počevši od pozicije **`110`**, **`110`**.

Poslednjom metodom, `clearRect(120, 120, 80, 80)`, prostor kvadratnog oblika od 80 piksela na kraju se uklanja iz sredine prethodnog pravougaonika.



Slika 10-1: *Prikaz platna i pravougaonici nacrtani pomoću koda iz listinga 10-3.*

Na slici 10-1 predstavljeno je ono što ćete videti kada se izvrši kôd iz listinga 10-3. Element `<canvas>` se prikazuje kao pomoćna mreža, čiji je koordinatni početak u gornjem levom uglu, a veličina zadata atributima. Pravougaonici se na platnu crtaju na poziciji zadatoj atributima `x` i `y`, jedan preko drugog, u skladu s redosledom iz izvornog koda. (Prvi se crta onaj pravougaonik koji je prvi naveden u izvornom kodu; drugi se crta preko prvog, itd.) Postoji metoda kojom se crtanje oblika može prilagođavati, ali o njoj ćemo govoriti kasnije.

Boje

Do sada smo podrazumevano koristili crnu boju. Boju zadajemo pomoću CSS sintakse i sledećih svojstava:

`strokeStyle` – Definiše boju linija oblika.

`fillStyle` – Definiše boju unutrašnjosti oblika.

`globalAlpha` – Ne zadaje boju, nego providnost za sve oblike nacrtane na platnu.

Listing 10-4: *Dodavanje boje.*

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

  canvas.fillStyle = "#000099";
  canvas.strokeStyle = "#990000";

  canvas.strokeRect(100, 100, 120, 120);
  canvas.fillRect(110, 110, 100, 100);
  canvas.clearRect(120, 120, 80, 80);
}
addEventListener("load", initiate);
```

Boje u listingu 10-4 deklarisane su pomoću heksadecimalnih brojeva. Možemo da koristimo i funkcije poput **`rgb()`**, ili čak da zadamo providnost oblika pomoću funkcije **`rgba()`**. Vrednosti koje se prosleđuju metodama stavljaju se u navodnike – na primer, **`strokeStyle = "rgba(255, 165, 0, 1)"`**.

Kada se boja zadaje pomoću ovih metoda, ona postaje podrazumevana i za ostale crteže.

Čak i kada se može koristiti funkcija **`rgba()`**, postoji svojstvo **`globalAlpha`** za definisanje nivoa providnosti. Njegova sintaksa je **`globalAlpha = vrednost`**, gde je **`vrednost`** broj između 0.0 (potpuna neprovidnost) i 1.0 (potpuna providnost).

Prelivi

Prelivi su danas veoma važni u programima za crtanje, pa ni API Canvas nije izuzetak. Kao i u CSS3, prelivi na platnu mogu biti linearni ili radijalni, i možemo da definišemo graničnike (engl. *stop points*) kako bismo kombinovali boje.

`createLinearGradient(`*`x1, y1, x2, y2`*`)` – Pomoću ove metode pravi se na platnu objekat s linearnim prelivom.

`createRadialGradient(`*`x1, y1, r1, x2, y2, r2`*`)` – Pomoću ove metode pravi se objekat s radijalnim prelivom, koji se na platnu realizuje pomoću dva kruga. Vrednosti predstavljaju poziciju središta svakog kruga i njegov poluprečnik.

`addColorStop(`*`pozicija, boja`*`)` – Pomoću ove metode zadaju se boje za pravljenje preliva. Pozicija je vrednost između 0.0 i 1.0, i određuje početak slabljenja date boje.

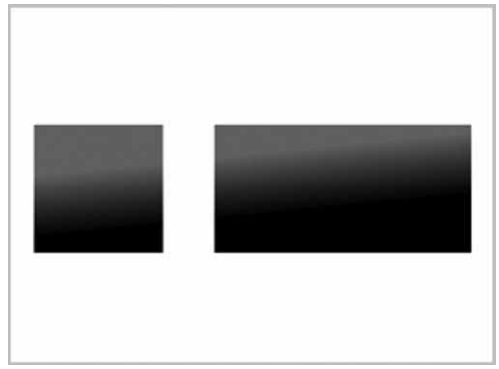
Listing 10-5: *Primena linearnog preliva na platno.*

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

  var grad = canvas.createLinearGradient(0, 0, 10, 100);
  grad.addColorStop(0.5, '#00AAFF');
  grad.addColorStop(1, '#000000');
  canvas.fillStyle = grad;

  canvas.fillRect(10, 10, 100, 100);
  canvas.fillRect(150, 10, 200, 100);
}
addEventListener("load", initiate);
```

U listingu 10-5 pravimo objekat s prelivom od položaja **`0, 0`** do položaja **`10, 100`**, pri čemu je preliv neznatno nagnut u levu stranu. Boje se zadaju pomoću metoda **`addColorStop()`**, a završni preliv se primenjuje na svojstvo **`fillStyle`**, kao obična boja.



Slika 10-2: *Linearni preliv za platno.*

Obratite pažnju na to da su pozicije preliva date u odnosu na platno, a ne u odnosu na oblike koje menjamo. To znači sledeće: ako pravougaonike pomerite na nov položaj na ekranu, njihovi prelivi će se promeniti.

> **Uradite sami** Radijalni preliv je sličan kao u jeziku CSS3. Linearni preliv u kodu iz listinga 10-5 pokušajte da zamenite radijalnim prelivom pomoću izraza kao što je **`createRadialGradient(0, 0, 30, 0, 0, 300)`**. Eksperimentišite i sa položajem pravougaonika da biste videli kako se preliv primenjuje.

Crtanje putanja

Pomoću metoda koje smo do sada proučavali crta se direktno na platnu, ali ne mora uvek biti tako. Oblike i slike često moramo obraditi u pozadini, a zatim rezultat poslati u kontekst u kome se iscrtava. API Canvas ima nekoliko metoda za generisanje putanja.

Putanja se može posmatrati kao linija neke skice koju pero treba da prati. Kada zadamo putanju, ona se šalje u kontekst i trajno iscrtava na platnu. Putanja može da sadrži različite vrste linija, kao što su prave linije, lukovi, pravougaonici i druge linije za crtanje složenih oblika.

Postoje dve metode za započinjanje i zatvaranje putanje:

`beginPath()` – Započinje opis novog oblika.

`closePath()` – Zatvara putanju, generišući pravu liniju iz poslednje tačke do početne. Ona se može izostaviti kada vam treba otvorena putanja, ili kada se putanja crta pomoću metode **`fill()`**.

Postoje i tri metode za crtanje putanje na platnu:

`stroke()` – Iscrtava putanju kao konturu.

`fill()` – Iscrtava putanju kao oblik punog tela. Kada se koristi ova metoda, putanja se ne mora zatvoriti metodom **`closePath()`**, jer se ona zatvara automatski pravom linijom od poslednje do prve tačke.

`clip()` – Zadaje novu površinu za isecanje (engl. *clipping area*) u datom kontekstu. Kada se kontekst inicijalizuje, površina za isecanje je cela oblast koju zauzima platno. Metoda **`clip()`** menja površinu za isecanje u nov oblik i tako pravi masku. Sve što se nalazi izvan maske neće se nacrtati.

Listing 10-6: *Započinjanje i završavanje putanje.*

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

  canvas.beginPath();
  // ovde dolazi putanja
  canvas.stroke();
}
addEventListener("load", initiate);
```

> **Podsetite se osnova** Komentari se mogu umetati u JavaScript kôd pomoću dve kose crte za komentare u jednom redu (**`// komentar`**), dok se za komentare u više redova koristi kombinacija kose crte i zvezdice (**`/* komentari */`**).

Kôd iz listinga 10-6 započinje putanju za kontekst platna i iscrtava je pomoću metode **`stroke()`**. Da bismo definisali putanju i nacrtali odgovarajući oblik, na raspolaganju nam je nekoliko metoda:

> **`moveTo(x, y)`** – Pomera pero na zadatu poziciju. Omogućava da započnemo ili nastavimo putanju iz različitih tačaka pomoćne mreže, čime se izbegavaju neprekidne linije.
>
> **`lineTo(x, y)`** – Generiše pravu liniju od tekuće pozicije pera do nove pozicije zadate atributima ***`x`*** i ***`y`***.
>
> **`rect(x, y, širina, visina)`** – Generiše pravougaonik ali, za razliku od ranije opisanih metoda, ovaj pravougaonik je deo putanje (ne crta se direktno na platnu). Atributi imaju istu namenu kao i u prethodnim metodama.
>
> **`arc(x, y, poluprečnik, početniUgao, završniUgao, smer)`** – Generiše lûk ili krug, čije se središte nalazi na koordinatama ***`x`*** i ***`y`***, dok su poluprečnik, uglovi i smer crtanja zadati ostalim atributima. Smer mora biti zadat kao **`true`** ili **`false`** (prva vrednost ukazuje na smer kretanja kazaljke na satu, a druga na obrnuti smer).
>
> **`quadraticCurveTo(cpx, cpy, x, y)`** – Generiše kvadratnu Bezjeovu krivu koja počinje od tekuće pozicije pera, a završava na poziciji zadatoj pomoću atributa ***`x`*** i ***`y`***. Atributi ***`cpx`*** i ***`cpy`*** zadaju kontrolnu tačku koja oblikuje krivu.
>
> **`bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)`** – Metoda slična prethodnoj, s tim što ima još dva atributa za generisanje kubne Bezjeove krive. Sada za oblikovanje krive u pomoćnoj mreži imamo dve kontrolne tačke deklarisane vrednostima ***`cp1x`***, ***`cp1y`***, ***`cp2x`*** i ***`cp2y`***.

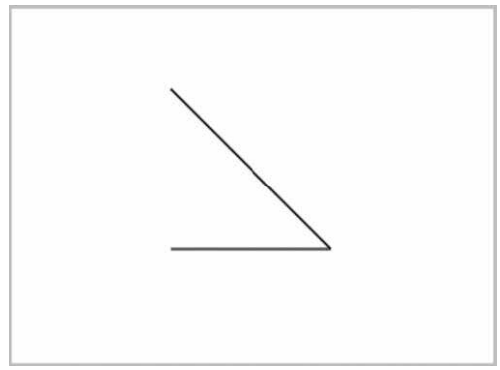
Razmotrimo jednostavnu putanju da bismo razumeli ove metode.

Listing 10-7: *Crtanje putanje.*

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

  canvas.beginPath();
  canvas.moveTo(100, 100);
  canvas.lineTo(200, 200);
  canvas.lineTo(100, 200);
  canvas.stroke();
}
addEventListener("load", initiate);
```

Preporučuje se da se početna pozicija pera zada odmah nakon započinjanja putanje. U skriptu iz listinga 10-7 prvo pomeramo pero na poziciju **100, 100**, a zatim od te tačke crtamo liniju do tačke **200, 200**. Pero je sada u tački **200, 200**, a naredna linija iscrtava se od nje do tačke **100, 200**. Na kraju se putanja iscrtava kao kontura pomoću metode **stroke()**.



Slika 10-3: *Otvorena putanja.*

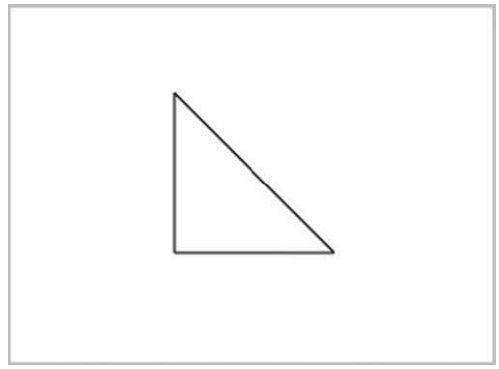
Na slici 10-3 je otvoreni trougao nacrtan pomoću skripta iz listinga 10-7. Ovaj trougao se raznim metodama može zatvoriti, ili čak ispuniti, kao u narednim primerima:

Listing 10-8: *Dovršavanje trougla.*

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

  canvas.beginPath();
  canvas.moveTo(100, 100);
  canvas.lineTo(200, 200);
  canvas.lineTo(100, 200);
  canvas.closePath();
  canvas.stroke();
}
addEventListener("load", initiate);
```

Metoda **closePath()** dodaje putanji pravu liniju od završne do početne tačke, i tako zatvara oblik.



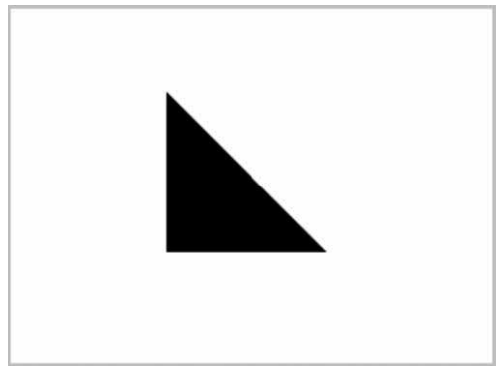
Slika 10-4: *Zatvorena putanja.*

Koristeći metodu **`stroke()`** na kraju naše putanje, nacrtali smo na platnu prazan trougao. Ako želimo popunjen trougao, upotrebićemo metodu **`fill()`**.

Listing 10-9: *Crtanje popunjenog trougla.*

```
function initiate(){
  var elem = document.getElementById('canvas');
  var canvas = elem.getContext('2d');
  canvas.beginPath();
  canvas.moveTo(100, 100);
  canvas.lineTo(200, 200);
  canvas.lineTo(100, 200);
  canvas.fill();
}
addEventListener("load", initiate);
```

Na ekranu je sada popunjen trougao. Metoda **`fill()`** automatski zatvara putanju, pa ne moramo više da koristimo metodu **`closePath()`**.



Slika 10-5: *Zatvorena putanja i popunjen trougao.*

Jedna od ranije predstavljenih metoda za crtanje putanje na platnu bila je metoda **clip()**. Pomoću nje se ništa ne crta, nego se pravi maska u obliku putanje i tako se bira šta će se crtati, a šta ne (sve što se nalazi izvan maske neće biti nacrtano).

Listing 10-10: *Upotreba trougla kao maske.*

```
function initiate(){
  var elem = document.getElementById('canvas');
  var canvas = elem.getContext('2d');
  canvas.beginPath();
  canvas.moveTo(100, 100);
  canvas.lineTo(200, 200);
  canvas.lineTo(100, 200);
  canvas.clip();

  canvas.beginPath();
  for(var f = 0; f < 300; f = f + 10){
    canvas.moveTo(0, f);
    canvas.lineTo(500, f);
  }
  canvas.stroke();
}
addEventListener("load", initiate);
```

Da bismo pokazali kako radi metoda **clip()**, u listingu 10-10 pravimo petlju **for** za generisanje horizontalnih linija na svakih 10 piksela. Linije se na platnu kreću sleva udesno, ali se prikazuju samo delovi koji su obuhvaćeni maskom u obliku trougla.



Slika 10-6: *Površina za isecanje.*

Naučili smo da crtamo putanju, i vreme je da se upoznamo sa alternativnim načinima izrade oblika. Znamo kako se generišu prave linije i kvadratni oblici. Za kružne oblike API nudi tri metode: **arc()**, **quadraticCurveTo()** i **bezierCurveTo()**. Prva je relativno jednostavna i može da generiše delimične ili pune krugove, kao u narednom primeru: