```
ctx.beginPath();
ctx.rect(150, 200, 100, 30);
ctx.fillStyle = "rgba(255,150,50,0.5) ";
ctx.fill();
```

Prostokąty są często spotykanymi figurami. Jeżeli chcesz narysować tylko jeden, a nie całą serię, istnieje jeszcze prostszy sposób na jego zadeklarowanie. Metoda `ctx.Rect()` przyjmuje cztery parametry, które definiują prostokąt wypełniany barwą z właściwości `ctx.fillStyle`:

```
ctx.fillStyle = "rgba(255,150,50,0.5) ";
ctx.fillRect(150, 200, 100, 30);
```

Podobny skrót można zastosować przy rysowaniu krawędzi prostokąta:

```
ctx.strokeStyle = "red";
ctx.strokeRect(150, 200, 100, 30);
```

Trzecim wariantem funkcji prostokąta jest `ctx.clearRect()`, która pobiera te same cztery parametry, ale zamiast rysować, czyści wskazany obszar. Wszystkie piksele w wyznaczonym polu zmieniają barwę na czarną, po czym ich kanał alfa jest ustawiany na 0, co czyni je przezroczystymi.

```
ctx.clearRect(150,  200,  100, 30);
```

Jeżeli chcesz wyczyścić całe płótno, podczas wywoływania metody po prostu podaj szerokość i wysokość elementu `canvas`. Tak naprawdę nawet przypisanie jednej z właściwości wymiarów ponownie jej samej powoduje reset komponentu.

```
canvas.width = canvas.width; // Element canvas został wyczyszczony.
```

Zmiana wartości jednej z właściwości wymiarów płótna automatycznie wywołuje jego wyczyszczenie. Uwaga! Zresetowanie elementu `canvas` w ten sposób doprowadzi do wyczyszczenia stylów wypełniania i obramowania oraz transformat i przycinanych ścieżek, o których opowiem więcej w dalszej części książki.

Zauważ, że funkcje `ctx.fillRect()`, `ctx.strokeRect()` i `ctx.clearRect()` są niezależne od API ścieżki i nie wymagają jej zdefiniowania przed ich wykorzystaniem. Nie wpływają również na dane ścieżki, więc można je bezpiecznie wywołać w trakcie ustawiania i rysowania kolejnych ścieżek.

Łuki i okręgi

Szybko zdasz sobie sprawę, że linie proste i prostokąty Ci nie wystarczą. Możesz też dodać segment łuku, używając w tym celu funkcji `arc()`:

```
ctx.arc(x, y, radius,  startAngle,  endAngle, ccw)
```

Funkcja ta tworzy fragment łuku w nowej podścieżce. Jest on generowany na podstawie wyimaginowanego okręgu o danym promieniu i środku w punkcie (x,y). Powstały fragment obwodu tego okręgu jest dodawany do podścieżki. Parametry `startAngle` i `endAngle` definiują, o jaki odcinek obwodu chodzi. Ostatni parametr, `ccw`, jest wartością `boolean` wskazującą kierunek rasteryzowania od kąta `startAngle` do kąta `endAngle`. Jeśli ma on wartość `true`, program używa kierunku zgodnego z ruchem wskazówek zegara; w innym przypadku używany jest kierunek przeciwny.

Uwaga: Kąty `startAngle` i `endAngle` wyrażane są w radianach. Liczba radianów dla pełnego okręgu jest równa 2π. Jeżeli wolisz pracować ze stopniami, możesz łatwo je przekonwertować na radiany, mnożące je przez $180/\pi$.
