Multimedia i interfejsy

Ćwiczenie 1 – video, canvas

Celem ćwiczenia jest poznanie nowych elementów wprowadzonych w HTML 5, do których należą m.in. video oraz canvas. Poniższy opis przedstawia sposób użycia tych elementów. Szczegółowe polecenia dotyczące ćwiczenia studenci otrzymają na zajęciach.

Element video

Element video pozwala na odtwarzanie na stronie pliku video. Posiada następujące atrybuty:

- autoplay – wyświetlanie filmu rozpocznie się automatycznie;
- controls – wyświetlone będą przyciski kontrolne (play, pause itd.);
- width, height – wymiary pola, w którym będzie wyświetlony film;
- loop – po zakończeniu film będzie automatycznie odtwarzany od początku;
- muted – wyłączenie dźwięku;
- poster – obraz, który będzie wyświetlany zanim rozpocznie się odtwarzanie pliku video;
- preload – sposób ładowania filmu; możliwe wartości: auto (ładowanie podczas ładowania strony), metadata (ładowanie tylko metadanych podczas ładowania strony), none (nic nie będzie ładowane podczas ładowania strony);
- src – adres URL pliku video.

Przykład użycia elementu video:

```
<video width = "320" height = "240" controls = "controls" src = "film.mp4">
</video
```

Przykład użycia elementu video wraz z elementem source w celu wskazania alternatywnych plików video:

```
<video width = "320" height = "240" controls = "controls">
        <source src = "film.mp4" type = "video/mp4">
        <source src = "film.ogg" type = "video/ogg">
        Przeglądarka nie obsługuje elementu video.
</video>
```

Element canvas

Element canvas umożliwia tworzenie grafiki (rysunków, wykresów, kompozycji zdjęć, animacji) przy użyciu języka JavaScript. Poniżej przedstawiono szablon dokumentu html pokazujący sposób użycia elementu canvas.

```
<!DOCTYPE html>
<html>
```

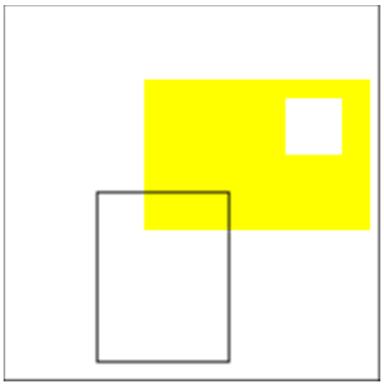
```
<head>
	<title>Multimedia i interfejsy - HTML5 </title>
	<script type = "text/javascript">
		function setup() {
			var canvas = document.getElementById('myCanvas');
			if (canvas.getContext) {			//jeśli przeglądarka wspiera canvas
				var ctx = canvas.getContext('2d');	//zmienna określająca kontekst graficzny
				//…tu można rysować…
			}
		}
	</script>
</head>
<body onload = "setup();">
	<canvas id = "myCanvas" width = "200" height = "200"></canvas>
			//atrybut id będzie wykorzystany do odwołania się do danego element w skrypcie
</body>
</html>
```

Dalsza część opisu przedstawia metody umożliwiające:

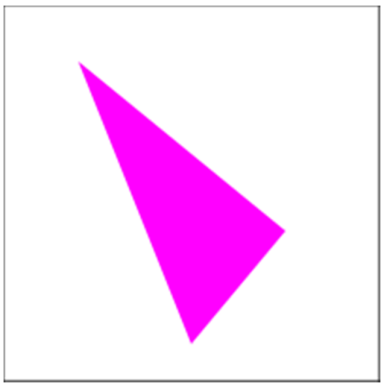
- rysowanie prostokątów, wielokątów, okręgów, krzywych,
- wypełnianie,
- wyświetlanie tekstu,
- transformacje (obracanie, skalowanie, przesuwanie),
- wyświetlanie obrazów.

Prostokąty

```
ctx.fillStyle = 'rgb(255,255,0)';		//rodzaj wypełnienia
ctx.fillRect(75,40,120,80);			//prostokąt wypełniony
ctx.strokeRect(50,100,70,90);			//prostokąt niewypełniony
ctx.clearRect(150,50,30,30);			//wyczyszczenie prostokąta
```

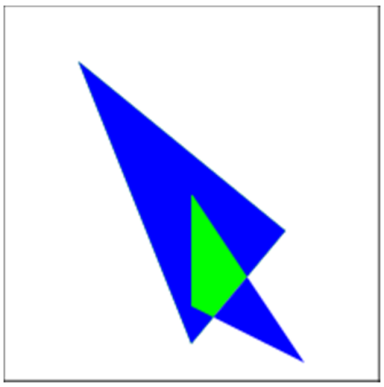


Wielokąty

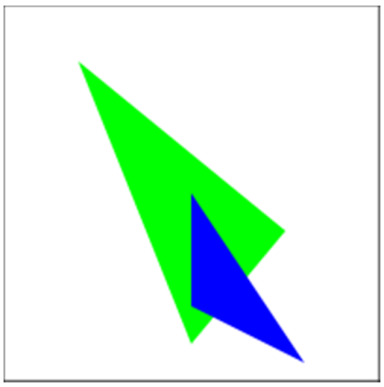


```
ctx.fillStyle = 'rgb(255,0,255)';
ctx.moveTo(40,30);                  //przesunięcie do danego punktu
ctx.lineTo(150,120);                //narysowanie linii do danego punktu
ctx.lineTo(100,180);
ctx.fill();                         //wypełnienie narysowanej figury
```

Ścieżki

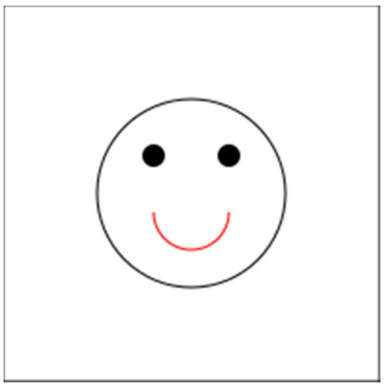


```
//chcemy narysować dwa nakładające się trójkąty (duży zielony i mały
// niebieski)
ctx.fillStyle = 'rgb(0,255,0)';     //zielony trójkąt
ctx.moveTo(40,30);
ctx.lineTo(150,120);
ctx.lineTo(100,180);
ctx.fill();
ctx.fillStyle = 'rgb(0,0,255)';     //niebieski trójkąt
ctx.moveTo(100,100);
ctx.lineTo(100,160);
ctx.lineTo(160,190);
ctx.fill();                         //funkcja fill dotyczy wszystkich
                                    //narysowanych dotychczas obiektów
//Jak to zmienić? Odpowiedź poniżej:
```



```
ctx.fillStyle = 'rgb(0,255,0)';
ctx.moveTo(40,30);
ctx.lineTo(150,120);
ctx.lineTo(100,180);
ctx.fill();
ctx.beginPath();                    //rozpoczęcie nowej ścieżki; funkcje
ctx.fillStyle = 'rgb(0,0,255)';     //fill lub stroke wywołane później
ctx.moveTo(100,100);                // będą dotyczyły tylko obiektów
ctx.lineTo(100,160);                // utworzonych od tego momentu
ctx.lineTo(160,190);
ctx.fill();
```

Koła, okręgi



```
                                        //głowa
ctx.arc(100, 100, 50, 0, Math.PI*2, false); //zdefiniowanie okręgu (współrzędne
                                        //środka, promień, kąt początkowy i
                                        //końcowy w radianach, kierunek)
ctx.stroke();                           //narysowanie okręgu

ctx.beginPath();                        //oczy
ctx.arc(80, 80, 6, 0, Math.PI*2, false);
ctx.arc(120, 80, 6, 0, Math.PI*2, false);
ctx.fill();                             //narysowanie koła

ctx.strokeStyle = 'rgb(255, 0, 0)';
ctx.beginPath();                        //usta
ctx.arc(100, 110, 20, Math.PI, 0, true);
ctx.stroke();
```

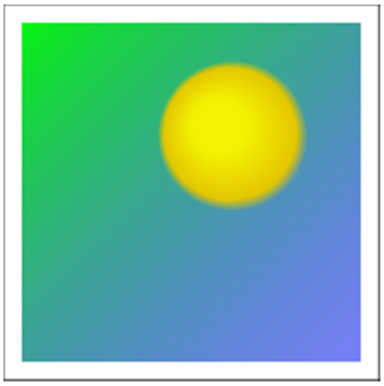
Krzywe Béziera

```
ctx.lineWidth = 5;                      //grubość linii
ctx.strokeStyle = "orange";             //kolor linii
ctx.fillStyle = "orange";               //kolor wypełnienia

ctx.beginPath();                        //antenka
ctx.moveTo(100,70);                     //punkt początkowy
ctx.quadraticCurveTo(90, 50, 115, 50);  //punkt kontrolny i końcowy
ctx.stroke();

ctx.beginPath();                        //beret
ctx.moveTo(50,120);                     //punkt początkowy
ctx.bezierCurveTo(20, 50, 180, 50, 150,120);  //punkty kontrolne i końcowy
ctx.fill();
```

Wypełnienie gradientowe



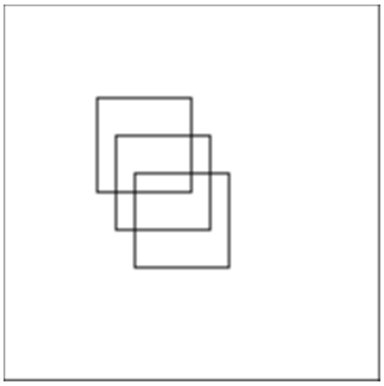
```
var grad = ctx.createLinearGradient(0, 0, 200, 200);      //gradient liniowy,
                                                           // x1,y1,x2,y2  określają
                                                           //kierunek zmian koloru
grad.addColorStop(0, 'rgb(0 , 255, 0)');     //ustalenie koloru w zadanym punkcie
                                              //obszaru objętego gradientem: 0 –
                                              //początek, 1-koniec  lub dowolna
                                              //wartość między 0 a 1; metodę
                                              //addColorStop można wywołać
                                              //dowolną liczbę razy
grad.addColorStop(1, 'rgba(0, 0, 255, 0.5)');
ctx.fillStyle = grad;
ctx.fillRect(10,10,180,180);

var radgrad = ctx.createRadialGradient(115,65,15,122,70,40);
                                              //gradient radialny; x1,y1,r1 – okrąg
                                              //początkowy; x2,y2,r2 – okrąg
                                              // końcowy
radgrad.addColorStop(0, '#F4F201');
radgrad.addColorStop(0.8, '#E4C700');
radgrad.addColorStop(1, 'rgba(228,199,0,0)');
ctx.fillStyle = radgrad;
ctx.fillRect(0,0,200,200);
```

Tekst


```
ctx.font = "bold 2em Comic Sans MS";
ctx.textAlign = 'center';
ctx.fillText('Hmmm...',100, 100);
```

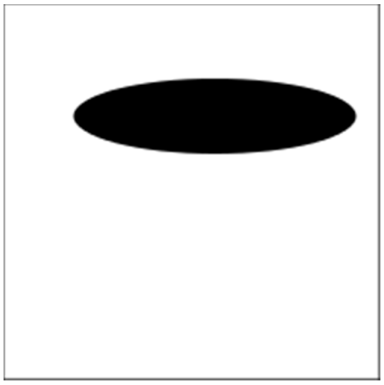
Przesuwanie



```
ctx.strokeRect(50,50,50,50);          //pierwszy prostokąt

ctx.translate(10, 20);
ctx.strokeRect(50,50,50,50);          //drugi przesunięty w stosunku do
                                      //pierwszego o wektor [10,20]
ctx.translate(10, 20);
ctx.strokeRect(50,50,50,50);          //trzeci przesunięty w stosunku do
                                      //drugiego o wektor [10,20]
```

Skalowanie

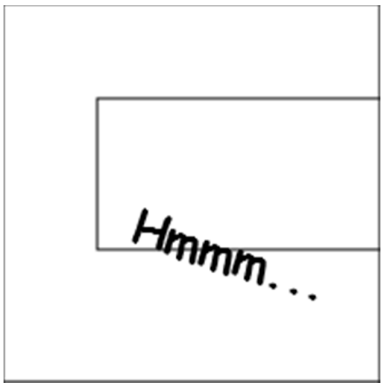


```
ctx.scale(1.5, 0.4);                  //zmiana proporcji (rozciągnięcie w
                                      // poziomie, ściśnięcie w pionie)
ctx.arc(75, 150, 50, 0, Math.PI*2, false);
ctx.fill();
```

Obracanie


```
ctx.rotate(0.3);                      //obrót o kąt podany w radianach
ctx.font = "bold 25px Comic Sans MS";
ctx.fillText('Hmmm...',100, 100);     //obrócony tekst
ctx.strokeRect(50, 50, 180, 80);      //obrócony prostokąt
```

Zapamiętywanie i odtwarzanie ustawień




```
ctx.save();                           //zapamiętanie stanu
ctx.rotate(0.3);                      //obrót o kąt podany w radianach
ctx.font = "bold 25px Comic Sans MS";
ctx.fillText('Hmmm...',100, 100);
ctx.restore():                        //odtworzenie stanu
ctx.strokeRect(50, 50, 180, 80);      //prostokąt nie będzie już obrócony
```

Obrazy

```
var img = new Image();                  //deklaracja nowego obiektu img
img.onload = function(){                //funkcja, która będzie wywołana
                                        //podczas zdarzenia onload dla
                                        //obiektu img
    ctx.drawImage(img,0,0,200,150);     //skopiowanie do canvas od pozycji
                                        //(0,0) obrazu z obiektu img i
                                        //przeskalowanie go do wymiarów
}                                       //200×150
img.src = 'zima.jpg';
```

```
var img = new Image();
img.onload = function(){
    ctx.drawImage(img,1500,700,1000,1000,0,0,200,200);
                                        //skopiowanie do canvas od pozycji
                                        //(0,0) fragmentu obrazu z obiektu
                                        //img o wymiarach 1000×1000
                                        // zaczynającego się od pozycji
                                        // 1500×700 i przeskalowanie go do
}                                       // wymiarów 200×200
img.src = 'zima.jpg';
```