Добавяне на интерактивност с HTML 5, jQuery и CSS 3

проф. д-р Асен Рахнев, докторант Мая Стоева

СЪДЪРЖАНИЕ

Използване на HTML5 Canvas API

I. Трансформации: Translate, Scale, Rotate, потребителски трансформации, огледални трансформации, Reset Transform, State Stack, Oval

II. Други ефекти: сянки (Shadows), Global Alpha, Clipping Region, операции

III. Работа с графични елементи: Image данни, конвертиране на цветове, картинки в сивата гама, зареждане на данни от URL, записване на изображения

IV. Canvas анимации: изтриване на платното, покадрова анимация (Animation Frames), линейни ефекти, ускоряване, полюшване (Oscillation), пускане и спиране на анимация

Използване на HTML5 Canvas API

```
<script>
  // get png data url
  var pngUrl = canvas.toDataURL();

  // get jpeg data url
  var jpegUrl = canvas.toDataURL('image/jpeg');

  // get low quality jpeg data url
  var lowQualityJpegUrl =
     canvas.toDataURL('image/jpeg', 0.2);
</script>
```

Използване на HTML5 Canvas API

За са вземем информацията за всеки пиксел от нашия canvas ползваме image data обекта и с метода **getImageData()** на canvas контекста и тогава имаме достъп до данните за пикселите чрез характеристиките на обекта image.

Използване на HTML5 Canvas API

```
<script>
  // взимаме image данните
  var imageData=context.getImageData(0,0,canvas.width,
     canvas.height);
  var data = imageData.data;

  // модифицираме ги

  // рисуваме отново върху платното
  context.putImageData(imageData, 0, 0);
</script>
```

Използване на HTML5 Canvas API-трансформации

```
<script>
  context.translate(x, y);
  // draw stuff
</script>
```

```
<script>
  context.scale(x, y);
  // draw stuff
</script>
```

```
<script>
  context.rotate(angle);
  // draw stuff
</script>
```

```
<script>
  context.transform(a, b, c, d, e, f);
  // draw stuff s custom transform
</script>
```

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

```
<script>
  context.transform(1 ,sy, sx, 1, 0, 0);
  // draw stuff naklanyane
</script>
```

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & s_x & 0 \\ s_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Използване на HTML5 Canvas API-трансформации

```
<script>
  // mirror horizontally
  context.scale(-1,1);

  // mirror vertically
  context.scale(1,-1);
</script>
```

```
<script>
  context.setTransform(1, 0, 0,
      1, 0, 0);
  // draw stuff
</script>
```

```
<script>
  context.save();
  // apply transforms
  // draw stuff
  context.restore();
</script>
```

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Използване на HTML5 Canvas API-ефекти

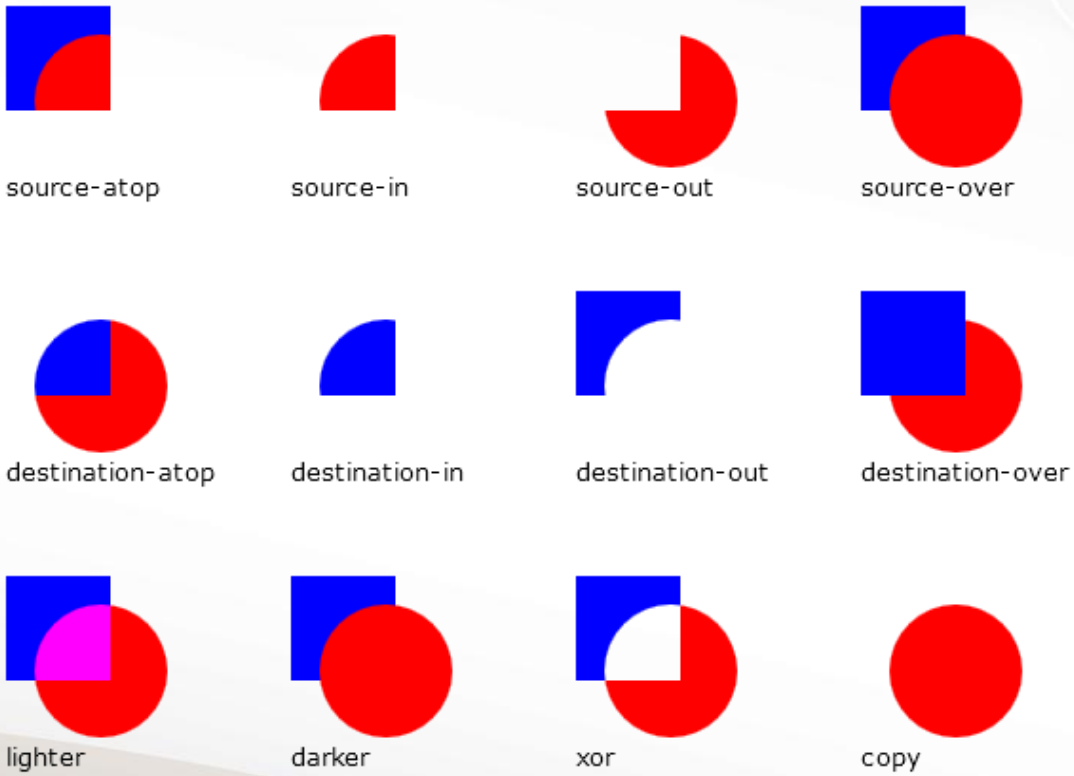
```
<script>
  context.shadowColor = 'red';
  context.shadowBlur = 20;
  context.shadowOffsetX = 10;
  context.shadowOffsetY = 10;
</script>

<script>
  context.globalAlpha = 0.6;
</script>
```

```
<script>
  context.save();
  // draw path here
  context.clip();
  // draw stuff here
  context.restore();
</script>
```

Използване на HTML5 Canvas API- Global Composite Operations

```
<script>
    context.globalCompositeOperation = 'destination-over';
</script>
```



Използване на HTML5 Canvas API-анимации

```
<script>
  context.clearRect(0, 0, canvas.width, canvas.height);
</script>
```

За да създадем линейно движение с HTML5 Canvas, можем да увеличаваме x или y, или и двете позиции на обекта за всеки кадър, според формулата за скорост.

За да създадем клатеща анимация с HTML5 Canvas, можем да използваме обикн. Хармонично трептене за позициониране на обекта за всеки кадър:

x(t) = отклонение * sin(t * 2PI / период) + x0

Използване на HTML5 Canvas API- анимации: спиране / пускане

За да пуснем HTML5 Canvas анимация, можем постоянно да извикваме нов кадър, а за да я спрем, просто не рекуестваме нов кадър.

http://kineticjs.com/

Използване на HTML5 Canvas API- прихващане на мишката

За да вземем релативните координати на мишката тук, можем да създадем getMousePos() метод, който връща координатите на мишката, на основата на позицията спрямо платното и получени на база getBoundingClientRect() метод на window обекта.