HTML5 Canvas

Ingo Köster

Diplom-Informatiker (FH)

HTML5 Canvas

- Das HTML5 Canvas erzeugt eine Zeichenfläche
- Mittels JavaScript können Grafiken und Bilder in das Canvas eingesetzt und animiert werden
- Der Inhalt des Canvas ist kein Teil der Webseite
- D.h. Mittels Javascript kann nicht auf die Objekte im Canvas zugegriffen werden
  - Mit SVG in einer HTML-Seite ist dies möglich

Canvas verwenden

› In HTML

```
<canvas id="myCanvas" width="300" height="300"></canvas>
```

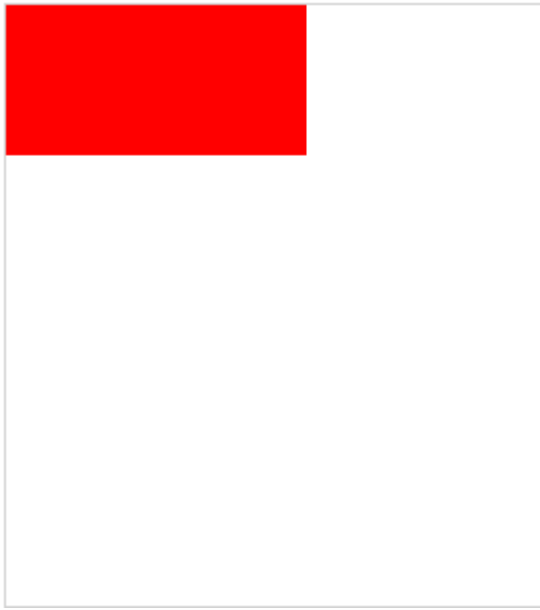
› Wichtig: Im HTML-Code die Höhe und Breite des Canvas-Elementes setzen und zur Laufzeit nicht mehr verändern

› Mehrere Canvas Elemente pro HTML-Seite sind erlaubt

Canvas verwenden

› In JavaScript

```
let myCanvas = document.getElementById("myCanvas");
let context = myCanvas.getContext("2d");
context.fillStyle = "red";
context.fillRect(0, 0, 150, 75);
```



› Erzeugt in der oberen linke Ecke des Canvas ein rotes Rechteck

› Breite 150 Pixel, Höhe 75 Pixel

Canvas verwenden

› Das Koordinatensystem des Canvas beginnt oben links (0,0)

› Beim Laden der Seite hat das Canvas keinen Hintergrund und ist nicht sichtbar

› Höhe und Breite des Canvas sind über das Objekt abrufbar

```
let canvas = document.getElementById("canvas");
let breite = canvas.width;
let höhe = canvas.height;
```

Rechtecke zeichnen

- `fillRect(x,y,width,height)`
- Zeichnet ein gefülltes Rechteck, beginnend bei den Koordinaten x,y und endend bei Breite und Höhe
  - Füllfarbe mit `fillStyle` angeben

- `strokeRect(x,y,width,height)`
- Zeichnet ein unausgefülltes Rechteck, beginnend bei den Koordinaten x,y und endend bei Breite und Höhe
  - Zeichenfarbe mit `strokeStyle` angeben

Farbeinstellungen

- `fillStyle & strokeStyle`
- Können
  - CSS Farbe
  - Gradient
  - oder Muster (aus Bilddatei)
- sein

- Standardfarbe ist schwarz
- Mittels `lineWidth` die Linienbreite für Stroke setzen
  - `context.lineWidth = 10;`

Linien zeichnen

› Zum Zeichnen einer Linie Start- und Endpunkt der Linie festlegen
  › `moveTo(x,y)` Startpunkt festlegen
  › `lineTo(x,y)` Endpunkt festlegen

› Mittels `stroke` die Linie zeichnen

› Beispiel

```
context.beginPath();
context.moveTo(0,0);
context.lineTo(200,200);
context.lineTo(100,100);
context.stroke();
```

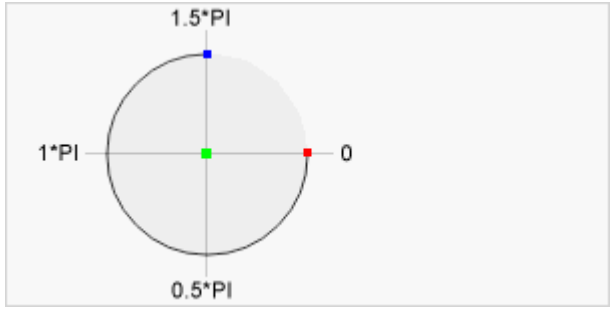
Kreise

› `arc(x,y,radius,start,stop)`

```
› context.beginPath();
  context.arc(100, 100, 40, 0, 1.5 * Math.PI);
  context.stroke();
```

› `start`
  › Startwinkel, in Radiant
  › 0 ist die 3 Uhr Position

› `stop`
  › Endwinkel, in Radiant



Canvas löschen

› `context.clearRect(0, 0, width, height)`

› Löscht alle Pixel im angegebenen Rechteck
  › Muss nicht das gesamte Canvas sein

Text

- Um Text in das Canvas zu schreiben werden `fillText` und `strokeText` verwendet
  - `fillText(text,x,y)`
  - `strokeText(text,x,y)`

- Zur Auswahl der Schriftart wird `font` verwendet

- Beispiel

```
let ctx = canvas.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
ctx.strokeText("Hello World",10,50);
```

Bild anzeigen - drawImage(image,x,y)

› In HTML

```
<img id="bild_id" src="…>
<canvas id="myCanvas" …>
```

› In JavaScript

```
let myCanvas = document.getElementById("myCanvas");
let context = myCanvas.getContext("2d");
let img = document.getElementById("bild_id");
context.drawImage(img, 10, 10);
```

Bild um 90 Grad drehen

› Beispiel:

```
let myCanvas = document.getElementById("myCanvas");
let context = myCanvas.getContext("2d");
let img = document.getElementById("bild_id");
context.rotate(90 * Math.PI/180);
context.drawImage(img, 10, 10);
```

› `context.rotate(90 * Math.PI/180)`

  › macht eine 90°-Drehung