

*Grafiken dynamisch erzeugen
in HTML5*



Canvas

kurz & gut

O'REILLY®

David Flanagan
Übersetzung von Lars Schulten

Inhalt

Vorwort	VII
Canvas-Tutorial	1
Linien zeichnen und Polygone füllen	6
Grafikattribute	10
Canvas-Dimensionen und -Koordinaten	13
Koordinatensystemtransformationen	15
Kurven zeichnen und füllen	22
Rechtecke	25
Farben, Transparenz, Verläufe und Muster	26
Strichattribute	31
Text	33
Clipping	35
Schatten	37
Bilder	40
Compositing	43
Pixelmanipulation	48
Treffererkennung	50
Canvas-Beispiel: Sparklines	52
Canvas-Referenz	55
Index	103

Canvas-Referenz

Dieser Teil des Buchs ist eine Referenz, die das `<canvas>`-Tag und die damit verbundenen Klassen dokumentiert. Die Referenz ist alphabetisch sortiert, und Methoden werden über ihren vollständigen Namen eingeordnet, der die Namen der Klassen einschließt, die sie definieren. Wenn Sie mehr über die `getContext()`-Methode erfahren wollen, müssen Sie beispielsweise `Canvas.getContext()` nachschlagen. Wollen Sie Näheres über die `arc()`-Methode wissen, schlagen Sie unter `CanvasRenderingContext2D.arc()` nach. Weil der Name dieser Klasse so lang ist, wird er in dieser Referenz mit »CRC« abgekürzt.

Der größte Teil dieses Kapitels dokumentiert Methoden des Canvas Rendering Context2Ds (unter dem Namen CRC), aber es werden auch Canvas, CanvasGradient, CanvasPattern, ImageData und TextMetrics behandelt.

Canvas

ein HTML-Element für skriptbasiertes Zeichnen

Eigenschaften

String `width`, `height`

Diese Eigenschaften spiegeln die `width`- und `height`-Attribute des `<canvas>`-Tags und geben die Maße des Canvas-Koordinatenraums an. Die Standardwerte für `width` und `height` sind 300 bzw. 150.

Wenn die Größe des Canvas-Elements nicht anderweitig in einem Stylesheet oder mit einem `style`-Attribut angegeben wird, bestimmen diese `width`- und `height`-Eigenschaften auch

den Platz, den das Canvas-Element auf dem Bildschirm einnimmt.

Wird eine dieser Eigenschaften gesetzt (sogar auch wieder auf den gleichen Wert), wird das Canvas geleert, und alle seine Grafikattribute werden auf die Standardwerte zurückgesetzt.

Methoden

`getContext()`

Gibt ein Kontext-Objekt zurück, über das Sie auf das Canvas zeichnen können. Übergeben Sie den String `2d`, erhalten Sie ein `CanvasRenderingContext2D`-Objekt für zweidimensionale Zeichnungen.

Übergeben Sie den String `webgl`, erhalten Sie ein `WebGLRenderingContext`-Objekt für 3-D-Darstellungen in Browsern, die das unterstützen. `WebGLRenderingContext` ist noch nicht standardisiert und wird in diesem Buch nicht dokumentiert.

`toDataURL()`

Gibt eine `data:`-URL zurück, die das Bild auf dem Canvas darstellt.

Beschreibung

Das Canvas-Objekt repräsentiert ein HTML-Canvas-Element. Es hat kein eigenes Verhalten, definiert aber eine API, die skriptbasierte clientseitige Zeichenoperationen ermöglicht. Sie können `width` und `height` direkt auf diesem Objekt angeben und mit `toDataURL()` ein Bild aus dem Canvas abrufen, aber die eigentliche Zeichen-API wird durch ein separates »Kontext-Objekt« gestellt, das von der Methode `getContext()` zurückgeliefert wird. Siehe CRC.

Das `<canvas>`-Tag wurde in Safari 1.3 eingeführt und wird mit HTML5 standardisiert. Es wird von allen aktuellen Versionen von Firefox, Safari, Chrome und Opera unterstützt. Es wird ebenfalls vom Internet Explorer 9 unterstützt und kann in früheren IE-Versionen mit der Open Source-Bibliothek `ExplorerCanvas` unter <http://code.google.com/p/explorercanvas/> emuliert werden.

Siehe auch

CRC

Canvas.getContext() liefert einen Kontext zum Zeichnen auf das Canvas

Überblick

Object getContext(String *contextID*)

Argumente

contextID

Dieses Argument gibt die Art der Zeichnung an, die auf dem Canvas erstellt werden soll. Übergeben Sie `2d`, um ein `CanvasRenderingContext2D`-Objekt zu erhalten, mit dem Sie zweidimensionale Zeichnungen ausführen können.

Rückgabewert

Ein Objekt, mit dem Sie in das Canvas-Element zeichnen können. Wenn Sie den String `2d` übergeben, ist das ein `CanvasRenderingContext2D`-Objekt für 2-D-Zeichnungen.

Beschreibung

Es gibt nur ein `CanvasRenderingContext2D`-Objekt pro Canvas-Element. Mehrfache Aufrufe von `getContext("2d")` liefern also dasselbe Objekt.

HTML5 standardisiert das `2d`-Argument für diese Methode und definiert keine weiteren gültigen Argumente. Ein anderer Standard, WebGL, für 3-D-Grafiken befindet sich in der Entwicklung. In unterstützenden Browsern können Sie dieser Methode den String `webgl` übergeben, um ein Objekt zu erhalten, das 3-D-Darstellung ermöglicht. Beachten Sie allerdings, dass das `CanvasRenderingContext2D`-Objekt der einzige Zeichenkontext ist, der in diesem Buch dokumentiert wird.

Siehe auch

CRC

Canvas.toDataURL() liefert ein Canvas-Bild als data:-URL

Überblick

String toDataURL()

String toDataURL(String *type*, *Parameter...*)

Argumente

type

Ein String, der den MIME-Typ für das zu nutzende Bildformat angibt. Wenn dieses Argument weggelassen wird, wird der Standardwert `image/png` verwendet. Das PNG-Format ist das einzige Format, das konforme Implementierungen unterstützen müssen.

Parameter...

Bei anderen Bildtypen als PNG können zusätzliche Argumente angegeben werden, die Kodierungsinformationen enthalten. Ist *type* `image/jpeg`, sollte das zweite Argument beispielsweise eine Zahl zwischen 0 und 1 sein, die die Qualitätsstufe des Bilds angibt. Aktuell sind keine weiteren Parameter standardisiert.

Rückgabewert

Ein String, der eine PNG-Darstellung der Canvas-Bitmap als `data:-URL` kodiert enthält.

Beschreibung

`toDataURL()` liefert den Inhalt der Canvas-Bitmap in einer URL-Form, die leicht in einem ``-Tag genutzt oder über ein Netzwerk übertragen werden kann.

Zur Vermeidung von Cross-Origin-Sicherheitslöchern funktioniert `toDataURL()` nicht mit `<canvas>`-Tags, die nicht »herkunftsfrei« sind. Ein Canvas ist nicht herkunftsfrei, wenn in es (direkt mit `drawImage()` oder indirekt über ein `CanvasPattern`) ein Bild gezeichnet wurde, das von einer anderen Quelle stammt als das Dokument, das das Canvas enthält.

Beispiel

```
// Den Inhalt des Canvas in ein img-Element kopieren
// und dieses Bild an das Dokument anhängen.
var canvas = document.getElementById("my_canvas");
var image = document.createElement("img");
image.src = canvas.toDataURL();
document.body.appendChild(image);
```

Siehe auch

`CRC.getImageData()`

Methoden`addColorStop()`

Gibt eine Farbe und eine Position für den Verlauf an.

Beschreibung

Ein `CanvasGradient`-Objekt repräsentiert einen Farbverlauf, der den `strokeStyle`- und `fillStyle`-Eigenschaften eines `CanvasRenderingContext2D`-Objekts zugewiesen werden kann. Die `createLinearGradient()`- und `createRadialGradient()`-Methoden des `CanvasRenderingContext2D`-Objekts liefern beide `CanvasGradient`-Objekte.

Haben Sie ein `CanvasGradient`-Objekt, nutzen Sie `addColorStop()`, um anzugeben, welche Farben an welchen Positionen des Verlaufs erscheinen sollen. Zwischen den von Ihnen angegebenen Positionen werden die Farben interpoliert, um einen stetigen Verlaufs zu erstellen. Wenn Sie keine Farbpositionen angeben, wird der Verlauf durchgängig schwarz.

Siehe auch

`CRC.createLinearGradient()`, `CRC.createRadialGradient()`

CanvasGradient.addColorStop() gibt eine Farbe auf dem Verlauf an**Überblick**

```
void addColorStop(float offset, String color)
```

Argumente

offset

Ein Fließkommawert im Bereich zwischen 0.0 und 1.0, der die Position zwischen Start- und Endpunkt des Verlaufs angibt. Eine Verschiebung von 0 entspricht dem Startpunkt und eine Verschiebung von 1 dem Endpunkt.

color

Gibt die an dieser Position anzuzeigende Farbe als CSS-Farbstring an. Farben an anderen Punkten auf dem Verlauf werden auf Basis dieser Farbe und der anderen Stoppfarben interpoliert.

Symbole

3-D-Grafiken 3

A

addColorStop() 30, 59
affine Transformationen 18
arc() 4, 22, 70
arcTo() 22, 71

B

beginPath() 4, 6, 8, 73
bezierCurveTo() 23, 73
Bilder 40
 zeichnen 68

C

c-Variable 5
Canvas-Maße 13
Canvas-Objekt 55
<canvas>-Tag 1, 56
 Internet Explorer, Einsatz
 in 2
CanvasGradient-Objekt 59
CanvasPattern-Objekt 60
CanvasRenderingContext2D-Objekt 2, 57, 70
 Eigenschaften 60
 Methoden 63, 65
clearRect() 25, 74
clip() 35, 74
Clipping 35
closePath() 7, 75

Compositing 43, 69
 Inkompatibilitäten 46
 lokal vs. global 47
copy-Compositing 44
CRC *siehe* CanvasRenderingContext2D-Objekt
createImageData() 48, 75
createLinearGradient() 29, 76
createPattern() 28, 77
createRadialGradient() 29, 78
CSS-Farbstring 26
CSS-Pixel 14

D

destination-over-Compositing
 44
drawImage() 40, 79
 Canvas, kopieren mit 43

E

ExplorerCanvas-Projekt 2

F

Farben, Verläufe und Muster 67
Farbnamen, HTML4-Standard
 26
Farbräume, von CSS3 unter-
 stützte 26
fill() 4, 6–7, 80
fillRect() 25, 81
fillStyle-Eigenschaft 26
fillText() 33, 82

G

`getContext()` 2, 11, 56–57
`getImageData()` 48, 83
Sicherheitsbeschränkungen
50
globalAlpha-Eigenschaft 27
globalCompositeOperation-Eigenschaft 44, 84
Grafikattribute 10, 13
Grafikzustand speichern 70
Grafikzustand, Werkzeuge zur
Verwaltung 12

H

HSL-Farbraum 27

I

ImageData-Objekt 100
`isPointInPath()` 50, 86

J

JavaScript V

K

Koch-Schneeflocken 19
Kontext-Objekt 3
Koordinatensystem 13
-transformationen *siehe*
Transformationen
Koordinatensystem und Trans-
formationen 69
Kurven zeichnen und füllen 22,
25

L

`lineCap`-Eigenschaft 32, 87
`lineJoin`-Eigenschaft 33, 88
`lineTo()` 6, 89
`lineWidth`-Eigenschaft 31

M

`measureText()` 89
Methoden zur Hinzufügung
und Verbindung von Punk-
ten 22
`miterLimit`-Eigenschaft 33, 90
MouseEvent-Umwandlung 50
`moveTo()` 6, 91
Muster- und Verlaufsfüllungen
28

N

Nonzero Winding-Regel 10

P

Pfad 4, 6, 22
erstellen und zeichnen 66
offene und geschlossene
Teilpfade 7
Pixel *siehe* CSS-Pixel
Pixelmanipulationsmethoden
48, 70
Polygone 8
`putImageData()` 48, 91

Q

`quadraticCurveTo()` 23, 92

R

Rechtecke zeichnen 25, 68
`rect()` 25, 93
`restore()` 12, 94
`rotate()` 16–17, 94

S

`save()` 12, 95
`scale()` 16–17, 95
Schatten 37, 69
`setTransform()` 15, 19, 96
`shadowBlur`-Eigenschaft 39

- shadowColor-Eigenschaft 38
- shadowOffsetX- und shadowOffsetY-Eigenschaften 38
- source-over-Compositing 44
- Sparklines 52, 54
- Strichbreite, Strichende und Strichverbindungen 68
- stroke() 6–7, 31, 97
- strokeRect() 25, 97
- strokeStyle-Eigenschaft 26
- strokeText() 33, 98

T

- Teilpfade 6
 - offene und geschlossene 7
- Text 33, 68
- textAlign- und textBaseline-Eigenschaften 34
- TextMetrics-Objekt 101
- toDataURL() 42, 56–57
- transform() 18, 99
- Transformationen 15, 20
 - Beispiel 19
 - Koordinatensystem und 69

- mathematische Erläuterung 17, 19
- Reihenfolge von 17
- translate() 16–17, 100
- Transparenz 26
 - Compositing-Operationen und 44
 - mit globalem Alpha 27
- Treffererkennung 50
- Tufte, Edward 52

V

- Verlaufsfüllung 28

W

- webgl (String) 3
- WebGL-API 3
- weiche Transparenz, Compositing-Operationen 44
- width- und height-Eigenschaften 55

Z

- Zeichenkontext 2