Grafiken dynamisch erzeugen in HTML5

Canvas

kurz & gut

David Flanagan
Übersetzung von Lars Schulten

O'REILLY®

Canvas

kurz & gut

David Flanagan

Deutsche Übersetzung
von Lars Schulten

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

Die Informationen in diesem Buch wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Verlag, Autoren und Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene Fehler und deren Folgen. Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Der Verlag richtet sich im Wesentlichen nach den Schreibweisen der Hersteller. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag GmbH & Co. KG
Balthasarstr. 81
50670 Köln
E-Mail: kommentar@oreilly.de

Copyright der deutschen Ausgabe:
© 2011 O'Reilly Verlag GmbH & Co. KG
1. Auflage 2011

Die Originalausgabe erschien 2010 unter dem Titel *Canvas Pocket Reference* bei O'Reilly Media, Inc.

Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

Übersetzung und deutsche Bearbeitung: Lars Schulten
Lektorat: Inken Kiupel
Korrektorat: Sibylle Feldmann
Produktion: Karin Driesen
Umschlaggestaltung: Karen Montgomery und Michael Oreal
Satz: Reemers Publishing Services GmbH, Krefeld, www.reemers.de,
Druck: fgb freiburger graphische betriebe; www.fgb.de

ISBN: 978-3-89721-597-9

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhalt

Vorwort

Dieses Buch dokumentiert die JavaScript-API zum Zeichnen von Grafiken in ein HTML-<canvas>-Tag. Es setzt voraus, dass Sie die Programmiersprache JavaScript kennen und zumindest grundlegend mit dem Einsatz von JavaScript in Webseiten vertraut sind. Kapitel 1 ist eine Einführung, die alle Canvas-Funktionen erklärt und anhand von Beispielen illustriert. Kapitel 2 beinhaltet eine Referenz zu den Canvas-bezogenen Klassen, Methoden und Eigenschaften.

Dieses Buch ist ein Auszug aus dem erheblich umfangreicheren Buch *JavaScript: Das umfassende Handbuch*; mein Verlag und ich waren der Meinung, dass das <canvas>-Tag eine so spannende HTML5-Komponente ist, dass es ein kompaktes eigenes Buch verdient. Da die Canvas-API recht überschaubar ist, kann sie in diesem kurzen Buch vollständig beschrieben werden.

Mein Dank gilt Raffaele Cecco, der das Buch und die Codebeispiele sorgfältig durchgesehen hat. Außerdem danke ich meinem Lektor, Mike Loukides, für seine Begeisterung für dieses Projekt, und Simon St. Laurent, der das Material vom Format für das »Umfassende Handbuch« in das »kurz & gut«-Format überführt hat.

Die Beispiele in diesem Buch können von der Webseite zum Buch heruntergeladen werden, auf der auch eventuelle Fehler aufgeführt werden, sollten solche nach Veröffentlichung des Buches entdeckt werden:

http://www.oreilly.de/catalog/canvasprger

Canvas-Tutorial

Dieses Buch erläutert, wie man mit JavaScript und dem HTML-<canvas>-Tag Grafiken in Webseiten zeichnet. Die Möglichkeit, komplexe Grafiken dynamisch im Webbrowser zu generieren, statt sie von einem Server herunterzuladen, stellt eine Revolution dar:

- Der Code, der auf Clientseite zur Erstellung der Grafiken genutzt wird, ist normalerweise erheblich kleiner als die Bilder selbst und spart damit eine Menge Bandbreite.
- Dass Aufgaben vom Server auf den Client verlagert werden, reduziert die Last auf dem Server und kann die Ausgaben für Hardware um einiges mindern.
- Die clientseitige Grafikgenerierung fügt sich gut in die Architektur von Ajax-Anwendungen ein, in denen der Server die Daten stellt und sich der Client um die Darstellung dieser Daten kümmert.
- Der Client kann Grafiken schnell und dynamisch neu zeichnen. Das ermöglicht grafikintensive Anwendungen (wie Spiele und Simulationen), die schlicht nicht machbar sind, wenn jeder Frame einzeln von einem Server heruntergeladen werden muss.
- Das Schreiben von Grafikprogrammen ist ein Vergnügen, und das <canvas>-Tag lindert für den Webentwickler die Pein, die die Arbeit mit dem DOM mit sich bringen kann.

Das <canvas>-Tag tritt nicht an sich in Erscheinung, sondern es erstellt eine Zeichenfläche im Dokument und bietet clientseitigem JavaScript eine mächtige API zum Zeichnen. Das <canvas>-Tag wird von HTML5 standardisiert, treibt sich aber schon deutlich länger herum. Es wurde von Apple in Safari 1.3 eingeführt und wird von Firefox seit Version 1.5 und Opera seit Version 9 unterstützt. Außerdem wird es von allen Versionen von Chrome unterstützt. Vom Internet Explorer wird es erst ab Version 9 unterstützt, kann in IE 6, 7 und 8 aber hinreichend gut emuliert werden.

Canvas im IE

Wenn Sie das <canvas>-Tag im Internet Explorer 6, 7 oder 8 einsetzen wollen, können Sie das Open Source-Projekt ExplorerCanvas unter *http://code.google.com/p/explorercanvas/* herunterladen. Entpacken Sie das Archiv und fügen Sie das excanvas-Skript im <head> Ihrer Webseiten mit einem bedingten Kommentar (Conditional Comment) für den Internet Explorer in folgender Form ein:

```
<!--[if lte IE 8]>
<script src="excanvas.compiled.js"></script>
<![endif]-->
```

Stehen diese Zeilen am Anfang Ihrer Webseiten, funktionieren <canvas>-Tag und elementare Canvas-Zeichenbefehle im IE. Radiale Verläufe und Clipping werden nicht unterstützt. Die Linienbreite wird nicht korrekt skaliert, wenn die x- und y-Dimensionen um unterschiedliche Beträge skaliert werden. Zusätzlich müssen Sie damit rechnen, beim IE weitere Rendering-Unterschiede zu sehen.

Große Teile der Canvas-Zeichen-API werden nicht im <canvas>-Element selbst definiert, sondern auf einem »Zeichenkontext-Objekt«, das Sie sich mit der getContext()-Methode des Canvas beschaffen. Rufen Sie getContext() mit dem Argument 2d auf, erhalten Sie ein CanvasRenderingContext2D-Objekt, über das Sie zweidimensionale Grafiken in das Canvas zeichnen können. Es ist wichtig, sich darüber

bewusst zu sein, dass das Canvas-Element und sein Kontext-Objekt zwei sehr verschiedene Objekte sind. Da der Klassenname so lang ist, verweise ich auf das CanvasRenderingContext2D-Objekt nur selten mit diesem Namen, sondern nenne es einfach das »Kontext-Objekt«. Und wenn ich von der »Canvas-API« spreche, meine ich damit in der Regel »die Methoden des CanvasRenderingContext2D-Objekts«. Da der lange Klassenname CanvasRenderingContext2D nicht gut auf diese schmalen Seiten passt, wird er außerdem im auf diese Einführung folgenden Referenzabschnitt mit CRC abgekürzt.

3-D-Grafiken in einem Canvas

Als dies geschrieben wurde, begannen Browserhersteller gerade damit, eine 3-D-Grafik-API für das `<canvas>`-Tag zu implementieren. Die entsprechende API wird als WebGL bezeichnet und ist eine JavaScript-Schnittstelle zur OpenGL-Standard-API. Ein Kontext-Objekt für 3-D-Grafiken erhalten Sie, wenn Sie der `getContext()`-Methode des Canvas den String `webgl` übergeben. WebGL ist eine umfangreiche und komplizierte Low-Level-API, die in diesem Buch nicht dokumentiert wird: Webentwickler werden wahrscheinlich eher auf WebGL aufgesetzte Hilfsbibliotheken nutzen als die WebGL-API selbst.

Ein einfaches Beispiel für die Canvas-API sehen Sie im folgenden Code, der ein rotes Rechteck und einen blauen Kreis in `<canvas>`-Tags schreibt und eine Ausgabe wie die generiert, die Sie in Abbildung 1-1 sehen.

```
<body>
Das ist ein rotes Rechteck:
<canvas id="square" width=10 height=10></canvas>.
Das ist ein blauer Kreis:
<canvas id="circle" width=10 height=10></canvas>.
<script>
// Das erste Canvas-Element und seinen Kontext abrufen
var canvas = document.getElementById("square");
var context = canvas.getContext("2d");
// Etwas in das Canvas zeichnen
context.fillStyle = "#f00";  // Die Farbe auf Rot setzen
```

```
context.fillRect(0,0,10,10); // Ein kleines Rechteck
                             // füllen

// Das zweite Canvas und seinen Kontext abrufen
canvas = document.getElementById("circle");
context = canvas.getContext("2d");
// Einen Pfad beginnen und ihm einen Kreis hinzufügen
context.beginPath();
context.arc(5, 5, 5, 0, 2*Math.PI, true);
context.fillStyle = "#00f";  // Die Füllfarbe auf Blau
                             // setzen
context.fill();              // Den Pfad füllen
</script>
</body>
```

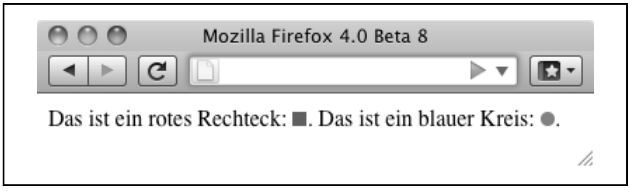

Abbildung 1-1: Einfache Canvas-Grafiken

Die Canvas-API beschreibt komplexe Figuren als »Pfade« von Linien und Kurven, die gezeichnet oder gefüllt werden können. Ein Pfad wird durch eine Folge von Methodenaufrufen definiert, wie beispielsweise die `beginPath()`- und `arc()`-Aufrufe aus dem vorangegangenen Code. Nachdem ein Pfad definiert ist, operieren andere Methoden wie `fill()` auf diesem Pfad. Verschiedene Eigenschaften des Kontext-Objekts wie `fillStyle` legen fest, wie diese Operationen ausgeführt werden. Die nächsten Unterabschnitte erläutern die folgenden Dinge:

- Wie man Pfade definiert und die Linie des Pfads zeichnet oder das Innere eines Pfads füllt.
- Wie man die Grafikattribute des Canvas-Kontext-Objekts setzt und abfragt und wie man den aktuellen Status dieser Attribute speichert und wiederherstellt.
- Canvas-Maße, das Standard-Canvas-Koordinatensystem und wie man dieses Koordinatensystem transformiert.