

Una chitarra in html5

Sviluppiamo un progetto grafico interattivo per esaminare le potenzialità di questo ambiente.

Lo standard html corrente è un salto quantico rispetto al precedente. Ci sono un buon numero di ragioni per affermarlo, la più importante è l'abbondanza di tag ricchi di semantica applicativa, come <menu> e <command>, di semantica rappresentativa, come <article> e <aside> e i tag <audio> e <video> per il supporto multimediale. Un'altra aggiunta carica di promesse è l'elemento <canvas>, che permette di disegnare in modo portatile su diversi browser e sistemi operativi. Questo mese facciamo una prima presentazione delle possibilità offerte dall'elemento <canvas>, utilizzando come pretesto il codice necessario per disegnare una tastiera di chitarra sullo schermo. In futuro potremo riciclare questa tastiera per produrre visualizzazioni interessanti per i musicisti, per esempio modi o scale.



Caratteristiche generali della canvas

Una canvas è un'area di video a disposizione per il disegno. Viene creata con un tag simile a quello che segue:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
Purtroppo il tuo browser non supporta html5, quindi non posso mostrarti questo disegno, che rappresenta la tastiera di una chitarra.
</canvas>
```

Come si fa di solito con estensioni del linguaggio, il testo alternativo per i browser storici si può mettere all'interno del tag <canvas>, in modo che appaia nei browser che non riconoscono il tag. Internet Explorer 9, Firefox, Opera, Chrome, e Safari supportano l'elemento <canvas>.

La lista dettagliata dei browser che supportano in vario grado l'elemento <canvas> si trova alla url caniuse.com/#search=canvas, con il grado di supporto, versione per versione. In ultima analisi la cronologia del supporto si traduce nella cronologia della resistenza all'adeguamento di Microsoft, dato che tutti gli altri browser hanno un supporto per <canvas> che risale alle prime generazioni.

Chrome, per esempio, ha il supporto base per la canvas da diciannove versioni, Safari da cinque, ma solo perché il ciclo di release è più lento. La base di codice, infatti, è la stessa, condivisa anche con la versione iOS di Safari e il browser di Android.

Quanto a Firefox e Opera, sono due alfieri dell'aderenza agli standard da sempre, per esempio, Firefox supporta la canvas da quindici versioni.

Internet Explorer 9 e 10 sono allineati con lo standard in modo soddisfacente e sono anche abbastanza veloci e leggeri (specialmente l'ultimo) da giustificare l'aggiornamento anche di un sistema un po' anziano.

A screenshot of the w3schools.com website. The page is titled 'HTML canvas arcTo() Method'. It includes a navigation bar with links to various web development topics. The main content area shows an example of using the arcTo() method to draw an arc between two tangents on a canvas. The example includes a JavaScript code snippet and a visual representation of the arc. The code is as follows:

```
var o=document.getElementById("myCanvas");
var ctx=o.getContext("2d");
ctx.beginPath();
ctx.moveTo(20,20); // Create a starting point
ctx.lineTo(100,20); // Create a horizontal line
ctx.arcTo(150,20,150,70,50); // Create an arc
ctx.lineTo(150,120); // Continue with vertical line
ctx.stroke(); // Draw it
```

Le pagine di w3schools dedicate alla canvas permettono di provare il codice interattivamente.

Funzioni della canvas

La canvas è molto ricca di opzioni. Per descriverle meglio le raggruppiamo per aree.

La prima contiene gli stili di riempimento, come il colore di default delle ombre o delle figure, come rettangoli e archi. Si possono usare immagini come pattern di riempimento e si possono creare pattern di riempimento ripetuti indefinitamente, come le campiture di un disegno tecnico, utilizzando delle bitmap.

Ogni figura può avere un'ombra di cui si può scegliere il colore, la sfocatura e lo spostamento rispetto alla figura.

Gli stili delle linee offrono la consueta ricchezza di variazioni, come un terminale butt, round o square, vale a dire tronco, arrotondato o quadrato, come è sin dai tempi dello X Toolkit del 1988. Anche le giunture fra le righe offrono le scelte abituali, come bevel, round e miter, cioè tronco, arrotondato o raccordato come una cornice.

I rettangoli possono essere disegnati e riempiti con un'unica chiamata, naturalmente lo stile di riempimento può essere un colore, un gradiente, un'immagine o un pattern.

Si possono creare percorsi, che includono linee, figure e curve quadratiche e di Bezier. I percorsi sono disegnati in un'unica soluzione, quando sono dichiarati completi, quindi si possono usare per creare figure complesse, come elementi di arredo, e disegnarli in un'unica soluzione.

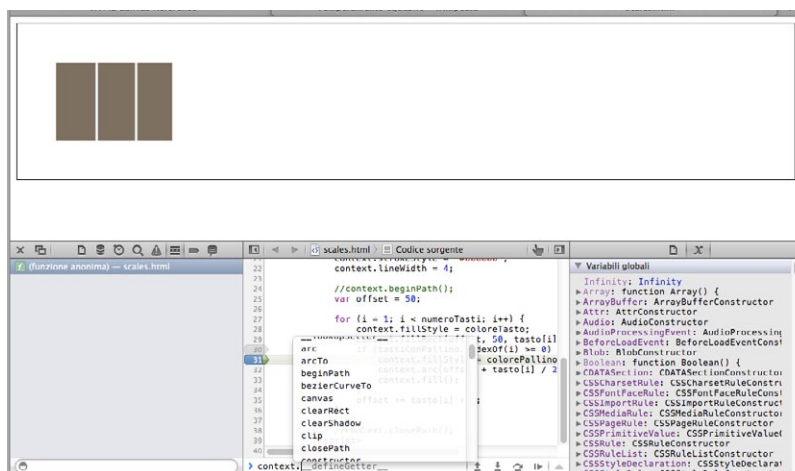
I comandi che riguardano il testo permettono di scegliere un font, una dimensione e disegnare del testo riempito o meno.

Infine, si possono comporre immagini usando diversi strati e combinandoli con la trasparenza, come negli editor di immagini basati su livelli.

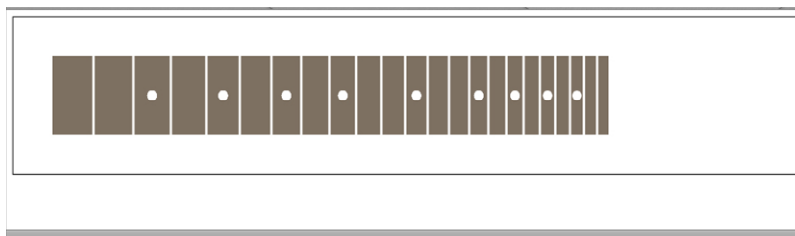
In sostanza, abbiamo un array di funzioni comparabili con buona parte dei toolkit grafici di base in circolazione.

Di sicuro si può realizzare un sistema di disegno evoluto con queste funzioni e, volendo, anche un cad bidimensionale, ma la presenza di funzioni per il disegno di immagini, tessiture e trasparenze, fa anche pensare a giochi abbastanza evoluti.

Chi vuole esplorare il ricco assortimento di funzioni può usare la pagina di riferimento su w3schools.com/tags/ref_canvas.asp. Ogni funzione ha un pulsante marcato *play*



Fermi in debug nella routine di disegno con Safari. Si vede il gran numero di membri del contesto grafico.



Ecco il risultato del nostro codice. Mancano pochi tocchi per diventare rifinito e professionale.

it, che permette di vedere un esempio di codice in azione e sperimentare con i parametri, insomma un ottimo campo da gioco per programmatori.

Cominciamo a sperimentare

La prima cosa da fare per creare un disegno è, ovviamente, creare una canvas, di dimensioni adatte. Nel nostro caso, poiché abbiamo in programma di creare una tastiera di chitarra, ci serve un'area di disegno piuttosto larga, cominciamo con 1000 per 200 pixel, poi, eventualmente, aggiusteremo il tiro.

Ecco l'inizio:

```
<canvas id="tastiera" width="1000"
height="200" style="border:1px
solid #000000;" />
```

A questo punto, la canvas è stata creata e ha il colore di fondo impostato al default, cioè nero trasparente, quindi non è visibile, a meno che non sia stato impostato un bordo colorato, come nel nostro caso, per delineare l'area di lavoro.

Cominciamo ora con il codice JavaScript che crea il disegno. Il primo passo è ottenere un riferimento all'elemento

<canvas> nel codice.

```
<script language="javascript">
var tastiera = document.
getElementById('tastiera');
var context = tastiera.
getContext('2d');
</script>
```

La prima riga, quindi, ottiene un riferimento alla canvas attraverso l'id. Si noti che abbiamo usato lo stesso nome per la variabile e l'id html, ma solo per economia di memoria, tanto non ci si può confondere.

Le coordinate della canvas iniziano in alto a sinistra, quindi il punto di coordinate (0,0) è quello in alto a sinistra, (0, 200) è in basso a sinistra e (1000, 200) è l'angolo in basso a destra.

Ottenuto il riferimento alla canvas, ci procuriamo un riferimento al suo contesto grafico con una chiamata al metodo *getContext* con il parametro "2d". D'ora in avanti manipoleremo il contesto grafico per disegnare e ci disinteresseremo dell'oggetto canvas chiamato *tastiera*. Ecco, per esempio, come possiamo impostare il contesto per disegnare linee spesse 4 pixel in colore nero.

```
context.strokeStyle =
```

```
"#000000";
context.lineWidth = 4;
```

Volendo creare una poligonale, potremmo iniziare con

```
context.beginPath();
```

proseguire con una serie di

```
context.moveTo(x, y);
context.lineTo(x', y');
```

Con *moveTo*, che sposta la penna e *lineTo* che disegna una riga. Esiste anche *arcTo*,

per disegnare archi. Per esempio, il codice seguente crea due linee raccordate da una curva

```
var c=document.
getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
// spostati a 20, 20
ctx.moveTo(20,20);
// disegna una linea orizzontale
ctx.lineTo(100,20);
// disegna un arco
ctx.arcTo(150,20,150,70,50);
// disegna una linea verticale
```

```
ctx.lineTo(150,120);
ctx.stroke();
```

La poligonale composta appare tutta insieme quando si esegue *ctx.stroke()*. Aggiungendo *ctx.closePath()*, potremmo creare una linea chiusa, che sarebbe riempita in accordo allo stile di riempimento impostato nel contesto nel momento in cui disegniamo.

E adesso la tastiera

Finalmente siamo al punto in cui possiamo disegnare la tastiera. La rappresenteremo come una collezione di rettangoli, di larghezza via via decrescente, secondo una progressione musicale.

Dedichiamo un attimo di attenzione alla logica con cui decrescono le dimensioni dei tasti. La regola generale è che il tasto consente di accorciare la corda, per aumentare in proporzione inversa la frequenza della nota emessa. Ai tempi di Pitagora, la musica si faceva calcolando i rapporti fra le note di un accordo usando frazioni intere, per esempio $2/3$ per il rapporto fra la lunghezza delle corde in un intervallo di quinta, il famoso power chord che domina il metal dai tempi di *Smoke on the water*.

Si può tentare di costruire un'accordatura in cui tutte le quinte sono giuste, ma questa accordatura, affascinata dal suono perfetto delle quinte, non riesce a stare entro i confini del 2:1 di un rapporto di ottava e non riesce a produrre accordature suonabili in tutte le tonalità.

La perfezione delle quinte, è stata storicamente un po' sacrificata aggiustando – cioè temperando – l'accordatura in modo da consentire di suonare in tonalità diverse. Bach lo dimostrò pubblicando un libro di pezzi scritti in tutte e dodici le tonalità, il *Clavicembalo ben temperato*.

L'accordatura temperata, il temperamento equabile, si basa su una progressione geometrica di ragione radice dodicesima di due. La lunghezza della corda che dà un mi, per esempio, viene divisa per radice dodicesima di due per avere la nota superiore, cioè un fa. Continuando a dividere per dodici volte per radice dodicesima di dodici, la corda si dimezza in modo esatto, perché stiamo dividendo per radice dodicesima di 2 alla dodicesima potenza. Le ottave sono perfette e le quinte sono

IL CODICE DI ESEMPIO

```
<html>
<body>
  <canvas id="tastiera" width="1000" height="200" style="border:1px solid
#000000;" />
  <script language="javascript">
    var tasto = new Array();
    var tastiConPallino = new Array (3, 5, 7, 9, 12, 15, 17, 19, 21);
    var scala = 900;
    var ratio = 0.94;
    var numeroTasti = 24;
    var coloreTasto = "#807060";
    var colorePallino = "#FFFFFF";

    for (i = 0; i < numeroTasti; i++) {
      tasto[i] = scala * (1 - ratio);
      scala -= tasto[i];
    }

    var tastiera = document.getElementById('tastiera');
    var context = tastiera.getContext('2d');

    context.strokeStyle = "#000000";
    context.lineWidth = 4;

    //context.beginPath();
    var offset = 50;

    for (i = 1; i < numeroTasti; i++) {
      context.fillStyle = coloreTasto;
      context.fillRect(offset, 50, tasto[i], 100);
      if (tastiConPallino.indexOf(i) >= 0) {
        context.fillStyle = colorePallino;
        context.arc(offset + tasto[i] / 2, 100, 6, 0, 2 * Math.PI);
        context.fill();
      }
      offset += tasto[i] + 3;
    }

    //context.closePath();
  </script>
</body>
</html>
```

leggermente fuori, ma in modo accettabile per buona parte degli orecchi. Rimandiamo i lettori desiderosi di approfondire alla pagina di Wikipedia it.wikipedia.org/wiki/Temperamento_equabile, che entra più in dettaglio rimanendo comprensibile ai più.

E ora si disegna

Assegniamo 900 dei 1.000 pixel alla nostra tastiera, quindi fissiamo una scala per la chitarra in cui la corda intera sarebbe 900 pixel. Calcoliamo la lunghezza dei tasti.

Diciamo quindi, che ogni tasto è largo quanto la lunghezza della corda corrispondente alla nota precedente meno la nota corrente.

La corda al primo tasto deve essere lunga quanto la scala, mentre al secondo tasto deve essere accorciata dividendo per radice dodicesima di due, un rapporto che chiamiamo *ratio*. Avremo, quindi

```
tasto = scala - scala * ratio
```

cioè

```
tasto = scala * (1 - ratio)
```

Reiteriamo il processo, ricalcolando la scala come la scala iniziale meno la lunghezza del primo tasto, e abbiamo il ciclo che calcola la lunghezza dei tasti:

```
for (i = 0; i < numeroTasti; i++) {
    tasto[i] = scala * (1 - ratio);
    scala -= tasto[i];
}
```

per il valore di *ratio* usiamo un'approssimazione: 0.94. L'imprecisione non è così grave, sul video.

Per disegnare, otteniamo un riferimento alla canvas e al suo contesto grafico

```
var tastiera = document.
getElementById('tastiera');
var context = tastiera.
getContext('2d');
```

Ora disegniamo i tasti, colorando un rettangolo di larghezza *tasto[i]* e altezza 100 a un offset orizzontale iniziale di 50 e un offset verticale fisso di 50.

```
var offset = 50;
```

```
for (i = 1; i < numeroTasti; i++) {
    context.fillRect(offset, 50,
    tasto[i], 100);
```

```
offset += tasto[i] + 3;
```

I tre pixel che aggiungiamo all'offset a ogni iterazione sono un espediente per lasciare uno spazio bianco che simula il tasto. Di nuovo, questo altererebbe l'intonazione in una vera tastiera. Cambiando il 3 in 2 avremmo tasti più sottili.

Qualche abbellimento

Per essere un po' più precisi, disegniamo anche i pallini bianchi che si trovano al terzo, quinto, settimo, nono e dodicesimo tasto.

Carichiamo i numeri "magici" in un array

```
var tastiConPallino = new Array (3,
5, 7, 9, 12, 15, 17, 19, 21);
```

E disegniamo un arco di due pi greco a una coordinata X di *offset + tasto[i] / 2*, cioè a metà del tasto in orizzontale. A metà del tasto in verticale, cioè a un offset di cinquanta più metà della larghezza della tastiera, che è cento, quindi una coordinata Y fissa a 100. Diamo al cerchio un diametro di 6 pixel ed ecco il codice

```
context.arc(offset + tasto[i] / 2,
100, 6, 0, 2 * Math.PI);
```

gli ultimi due parametri sono l'inizio e la fine dell'arco, cioè zero e due pi greco. Ecco il codice completo del ciclo di disegno dei tasti, completato con la gestione di due colori di riempimento, uno per i tasti e uno per i pallini.

```
for (i = 1; i < numeroTasti; i++) {
    context.fillStyle =
    coloreTasto;
    context.fillRect(offset, 50,
    tasto[i], 100);
    if (tastiConPallino.indexOf(i)
    >= 0) {
        context.fillStyle =
        colorePallino;
        context.arc(offset +
        tasto[i] / 2, 100, 6, 0, 2 * Math.
        PI);
        context.fill();
    }
    offset += tasto[i] + 3;
}
```

Con questo codice abbiamo completato l'escursione della parte di *listato* che richiede chiarimenti, il resto dovrebbe essere chiaro da sé. •

ABBONATI SUBITO!

PC PROFESSIONALE
8,5 GB DI SOFTWARE UTILE
in regalo DVD DOPPIO
Office 2013
Sotto esame la versione definitiva della nuova suite di produttività Microsoft
Fotocamera
Sei compatte evolute che non fanno rimpiangere le reflex
Senza carta: tre scanner per documenti, veloci e leggeri, ideali per casa e piccolo ufficio
Kobo Arc
L'alternativa economica all'iPad Mini
TRUCCHI E TECNOLOGIE PER SFRUTTARE E CONDIVIDERE TUTTE LE PERIFERICHE E CONTENUTI MULTIMEDIALI IN CASA
Internet ovunque
Condivisione dei contenuti
Dina per distribuire audio/video
Nas: un server tuttofare
Accedere da remoto alla Lan
È VELOCE WINDOWS 8?
Quanto influisce sulle prestazioni l'upgrade tra Windows 7 e Windows 8. La prova a confronto su due piattaforme hardware diverse.

CARTACEO + DIGITALE

60%

SCONTO PER DUE ANNI

Solo 66,00 euro invece di 165,60

24 numeri + **edizione digitale**

55%

SCONTO PER UN ANNO

Solo 37,00 euro invece di 82,80

12 numeri + **edizione digitale**

DIGITALE

64%

SCONTO PER UN ANNO

Solo 29,99 euro invece di 82,80

12 numeri da sfogliare sul tuo tablet o sul Pc

informazioni su www.abbonamenti.it