Guida HTML5: il Canvas - Parte 1

Come accennato nell'introduzione, uno degli obiettivi dell'HTML5, è facilitare l'implementazione di **applicazioni interattive**. Una delle più importanti tra queste è **canvas**: una **tela virtuale** attraverso la quale potrete liberare l'artista che è in voi!

A differenza degli altri elementi introdotti **nell'HTML5**, canvas richiede il supporto di **JavaScript**; ciò lo rende principalmente uno strumento di programmazione che sdogana per la prima volta i limiti imposti nei vecchi documenti web.

Inizieremo disegnando **elementari linee** per poi arrivare, nel corso delle prossime lezioni, ad **applicazioni grafiche** più avanzate: **animazioni**, **giochi**, **grafici dinamici**.

Fino a poco tempo fa era impensabile poter creare qualcosa del genere senza l'aiuto di plugin installati nel browser come Flash.

Ma bando alle chiacchiere, iniziamo.

Linee dritte

Il primo passo è inserire nel **body** del nostro documento questo semplice tag:

Qui specifichiamo **altezza**, **larghezza** e, più importante tra tutti, **l'ID** che ci servirà come riferimento nel **codice JavaScript**.

Fate attenzione a non incorrere nell'errore di dichiarare le proprietà width ed height all'interno del foglio CSS anziché nel markup, come specificato nell'esempio appena fatto!

Se provaste in questo momento a visualizzare la vostra pagina html vi trovereste davanti ad un semplice documento bianco.

Dov'è finito il tag che abbiamo appena inserito? Per renderlo visibile basta attribuire nel nostro foglio di stile la seguente regola:

Il terzo step fondamentale consiste nell'utilizzare del codice **JavaScript** che si agganci **all'ID** del canvas:

e, successivamente, utilizzare il metodo **getContext** dell'oggetto canvas per specificare il contesto **bidimensionale**:

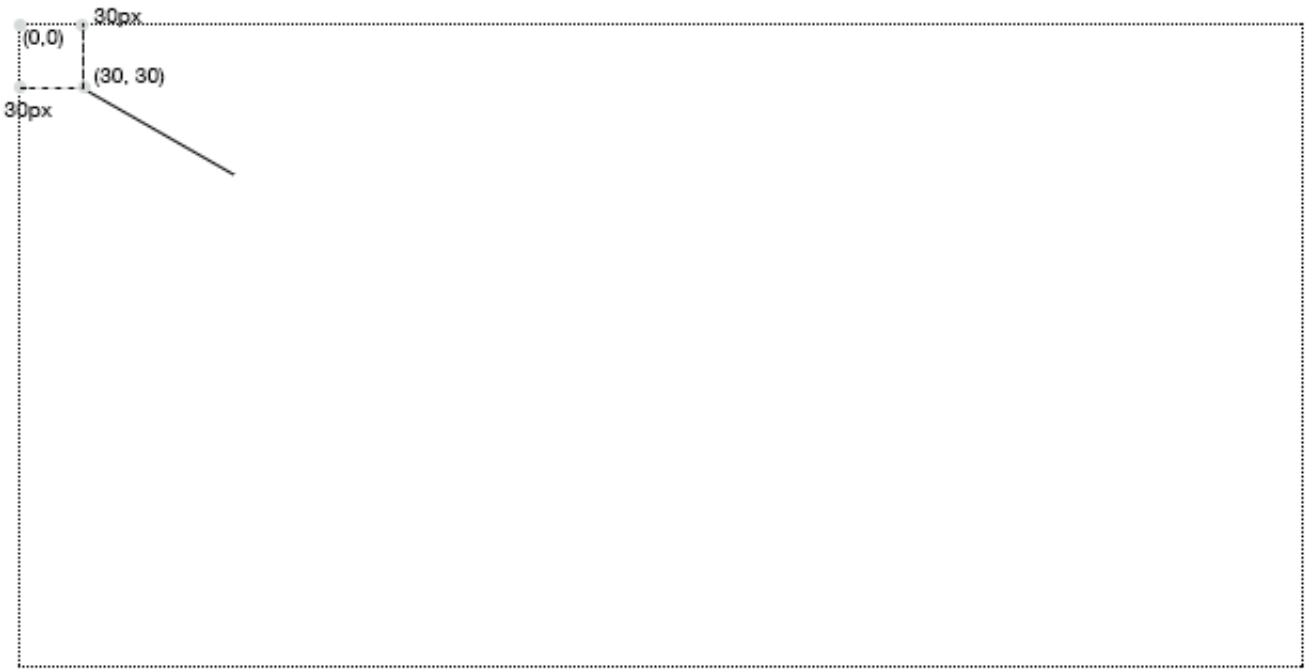
A questo punto sorgerà spontanea la domanda: "se abbiamo specificato 2d, esiste anche un contesto di **disegno tridimensionale**?".

Calmi, calmi.

La domanda è complessa ed una risposta esauriente sull'utilizzo di contesti avanzati, come il **WebGL**, potrà essere data solo alla fine di questo percorso!

Prima di poter proseguire nel disegno occorre, inoltre, capire il sistema di coordinate a cui fanno riferimento i movimenti su canvas. In poche parole il **punto iniziale**, ovvero la **coordinata (0,0)** corrisponderà **all'angolo superiore sinistro** del vostro spazio di lavoro; partendo da quel punto potrete misurare le vostre coordinate in **pixel**.

Per capire meglio il concetto, fate riferimento a quest'immagine:



Finalmente possiamo terminare il compito che c'eravamo prefissi nell'intestazione: disegnare semplici linee dritte. Le azioni da utilizzare sono tre:

- **moveTo** - per settare il **punto di partenza** dal quale disegnare;
- **lineTo** - per definire il **punto di arrivo** al quale arriverà la nostra linea;
- **stroke** - per **disegnare** realmente il **segmento** precedentemente definito;

Arrivati a questo punto, il nostro codice dovrebbe apparire così:

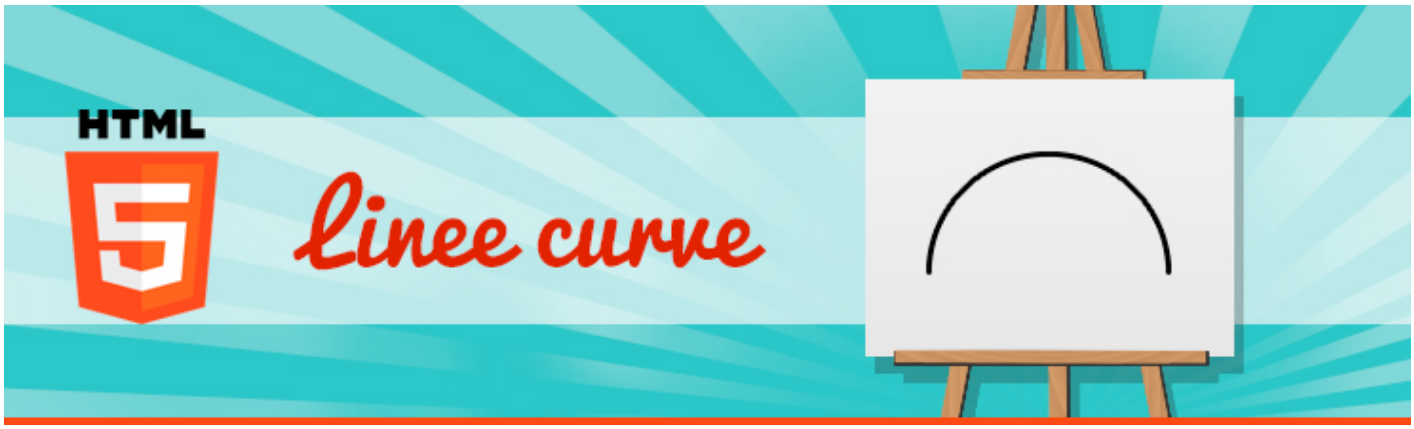
Mentre il **risultato visivo** dovrebbe essere:

Ovviamente abbiamo la possibilità di **personalizzare** un po' le linee che andremo a disegnare.

Ad esempio, possiamo scegliere il **colore** e la **larghezza**:

In questo esempio il colore è stato espresso in **RGB** ma è possibile utilizzare anche usuali codici **HEX**; nel nostro esempio #06B092.

Linee curve



Ma nella vita non esistono solo linee dritte, giusto? Allora vediamo come disegnare **linee curve** attraverso le quattro fondamentali funzioni: **arc**, **arcTo**, **bezierCurveTo**, e **quadraticCurveTo**.

Il metodo più intuitivo e facile da imparare inizialmente è **arc** che si occupa di disegnare un **arco basato su una porzione di cerchio**. Ad esempio:

Analizzando il codice, possiamo capire che:

- le variabili **centerX** e **centerY** determinano il centro del nostro ipotetico cerchio;
- la variabile raggio imposta il raggio in **pixel**;
- le variabili **inizioAngolo** e **fineAngolo** impostano l'inizio e la fine dell'arco che vogliamo ottenere;
- la funzione **context**.**arc** utilizza tutte le variabili finora citate per tracciare il nostro arco;

Visivamente avremo:

Gli altri metodi, *arcTo*, *bezierCurveTo*, e *quadraticCurveTo* (ovvero le cosiddette '**curve per punti** ') possono risultare un po' più ostici ai meno esperti. L'esempio ideale per iniziare ad approcciare alle curve per punti è, indiscutibilmente, la **curva di Bézier**.

Per chi non conoscesse la curva di Bézier, presente ormai in qualsiasi programma grafico, può approfondire l'argomento su Wikipedia. Ma vediamo subito come crearne una:

Analizziamo, anche in questo caso, il codice:

- le variabili **punto1x**, **punto1y** contengono le coordinate, in pixel, del primo punto mentre le variabili **punto2x**, **punto2y** quelle del secondo;
- le variabili **ultimoPuntoX** e **ultimoPuntoY** contengono, come intuibile, le coordinate del punto finale;
- la funzione **context.bezierCurveTo** utilizza le quattro variabili precedentemente descritte per tracciare la linea curva;

Ovviamente tracciare curve di Bézier in canvas può diventare complesso, senza avere riferimenti grafici immediati. Per risolvere questo problema, potete utilizzare questi due ottimi strumenti: link1 e link2.

Conclusione

Ricapitolando, abbiamo appreso come inserire un elemento canvas in un documento HTML e come, attraverso poche righe di **JavaScript**, **disegnare linee dritte e curve**. Chi non ha conoscenze di JavaScript può aver trovato difficoltà nel seguire alcune porzioni di codice; in tal caso consiglio di studiare i rudimenti di questo linguaggio prima di approcciare, nuovamente, al canvas.

Nel corso del prossimo articolo vedremo come **usare comandi canvas più dettagliati**, come:

trasparenza, **ombre**, **controllo** e **trasformazione delle immagini**.

Ma, nel frattempo, dopo avervi mostrato alcune delle potenzialità di canvas, pensate che esso sancisca di fatto la fine dell'utilizzo di Flash in ambito web?

Preferite continuare a sviluppare animazioni, grafici e giochi in **ActionScript** o pensate che **canvas** e il suo JavaScript possano svolgere bene il loro ruolo?

Inoltre, **Flash** sarebbe comunque giunto al capolinea, se non ne avesse ostacolato la distribuzione sui propri devices?

E' possibile scaricare il file .zip di questa lezione a questo link.

GUIDA HTML5: GLI ARTICOLI
1) Guida HTML5:Introduzione
2) Guida HTML5: la prima pagina
3) Guida HTML5: la struttura
4) Guida HTML5: Immagini e outlines
5) Guida HTML5: nuovi elementi semantici
6) Guida HTML5: i form – Parte 1
7) Guida HTML5: i form – Parte 2
8) Guida HTML5: i form – Parte 3
9) Guida HTML5: i form – Parte 4
10) Guida HTML5:i tag audio e video – parte 1
11) Guida HTML5:i tag audio e video – parte 2
12) Guida HTML5: I player video
13) HTML5: Il Canvas – Parte 1
14) Guida HTML5: Il Canvas - Parte 2
15) HTML5: Il Canvas - Parte 3
16) HTML5: Il Canvas - Parte 4
17) HTML5: Web storage