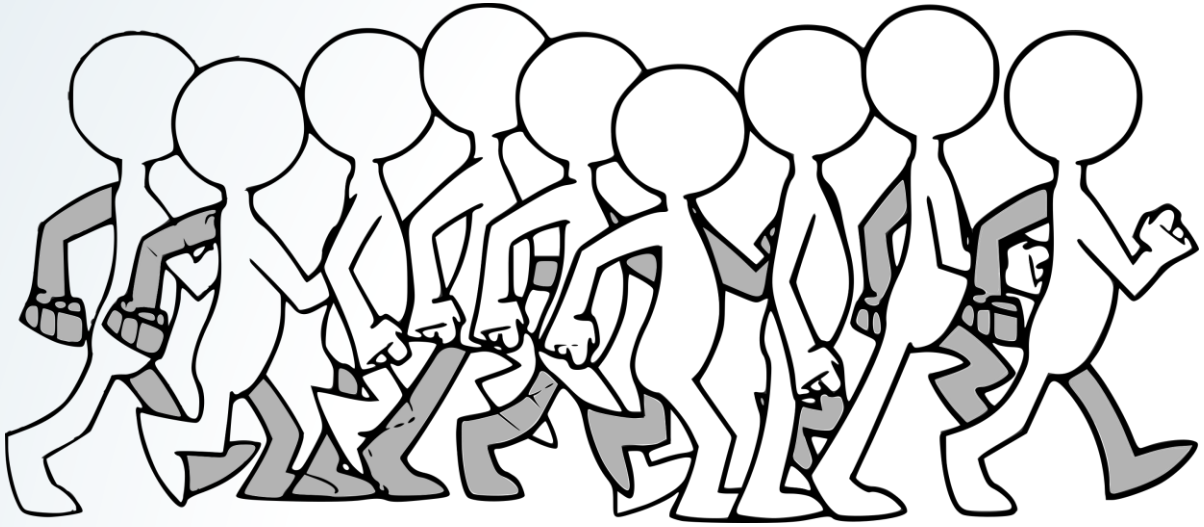
Animazioni in SVG



a cura di Massimiliano Aprea

development innovation

Argomenti trattati oggi:

- Anatomia di un SVG
- Fondamenti di GreenSock
- Timeline di GreenSock
- Animazioni interattive
- Possibili applicazioni

development innovation

IL CIELO E' IL LIMITE

Il libro di Frank Thomas e Ollie Johnston *"Disney Animation: The Illusion of Life"* (Disney Editions) inizia con questa citazione:

"Animation can explain whatever the mind of man can conceive"

Questa è la vera forza dell'animazione: ogni cosa può essere realizzata e *"l'unico limite è il cielo"*.

Anatomia di un SVG

Gli SVG stanno diventando sempre più popolari come mezzo per fornire immagini sul WEB. Il motivo risiede nel nome

DEFINIZIONE

SVG: *Scalable Vector Graphics*

Scalable: scalabili dimensionalmente, che in un'epoca con dispositivi di dimensioni sempre più diverse è un vantaggio enorme per lo sviluppo

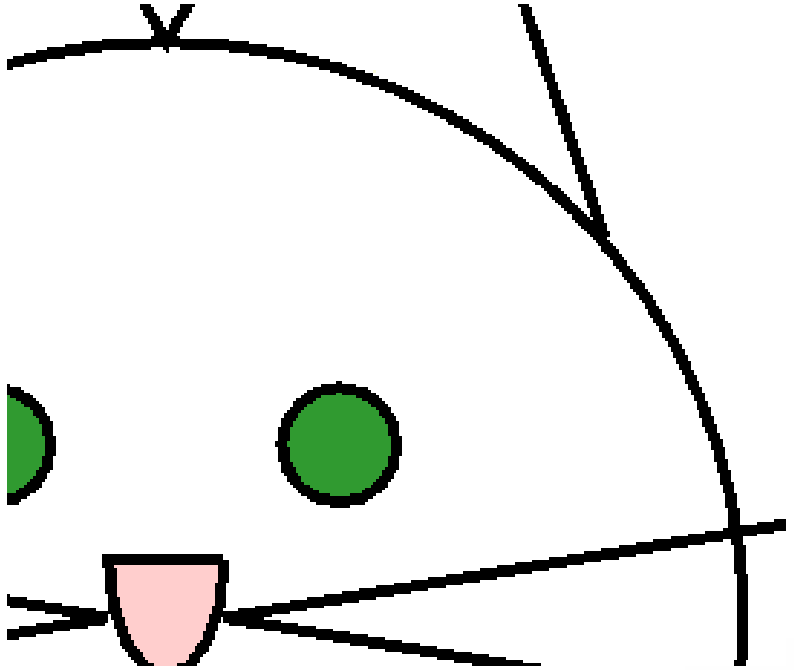
Vector: (invece che raster) significa che, poiché sono disegnati matematicamente, gli SVG tendono ad avere prestazioni maggiori e dimensioni minori

K2 development innovation

Confronto tra immagini *raster* e *SVG*

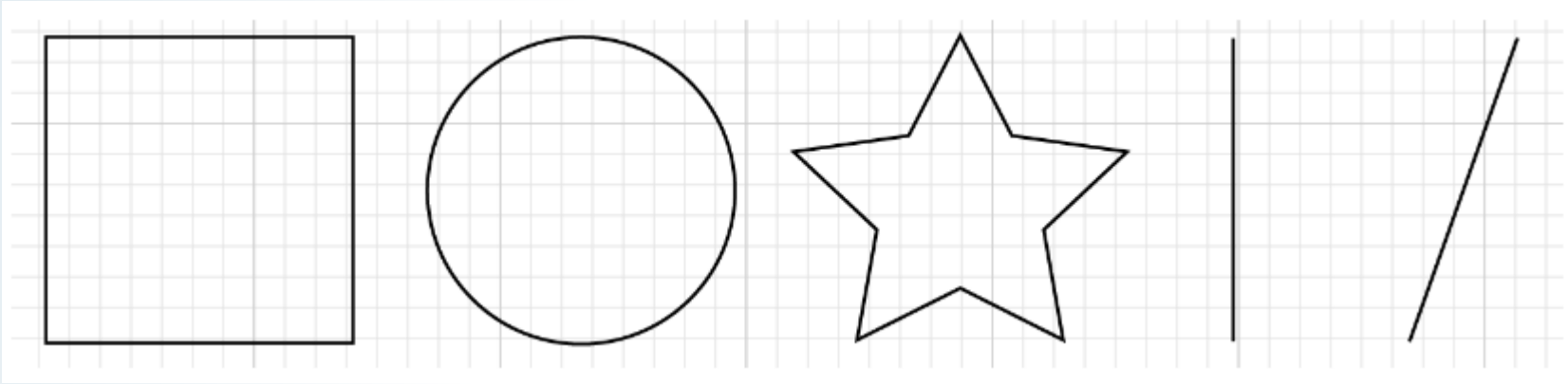
Raster

SVG



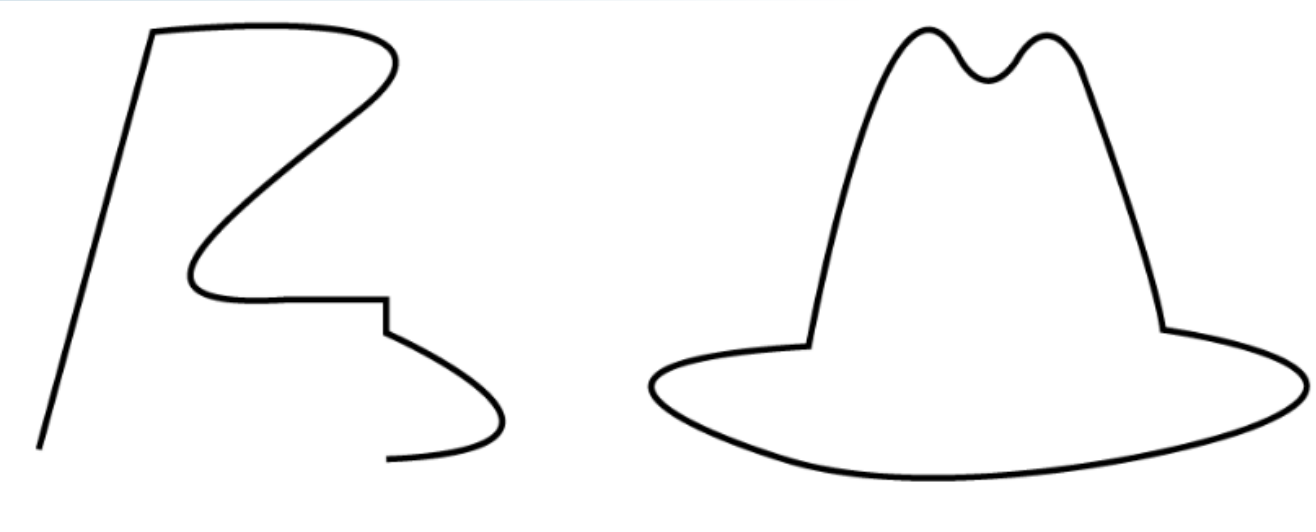
Sintassi DOM di un SVG

```
<svg x="0px" y="0px" width="450px" height="100px" viewBox="0 0 450 100">
 <rect x="10" y="5" fill="white" stroke="black" width="90" height="90"/>
 <circle fill="white" stroke="black" cx="170" cy="50" r="45"/>
 <polygon fill="white" stroke="black" points="279,5 294,35 328,40 303,62
  309,94 279,79 248,94 254,62 230,39 263,35"/>
 <line fill="none" stroke="black" x1="410" y1="95" x2="440" y2="6"/>
 <line fill="none" stroke="black" x1="360" y1="6" x2="360" y2="95"/>
</svg>
```



SVG responsivi, raggruppamento e disegno di percorsi

```
<svg viewBox="0 0 218.8 87.1">
 <g fill="none" stroke="#000">
   <path d="M7.3 75L25.9 6.8s58.4-6.4 33.5 13-41.1 32.8-11.2 30.8h15.9v5.5s42.6
     18.8 0 20.6" />
   <path d="M133.1 58.2s12.7-69.2 24.4-47.5c0 0 4.1 8.6 9.5.9 0 0 5-10 10.4.9 0
     0 12.2 32.6 13.6 43 0 0 39.8 5.4 15.8 15.4-13.2 5.5-53.8
     13.1-77.4 5.9.1 0-51.9-15.4 3.7-18.6z" />
 </g>
</svg>
```

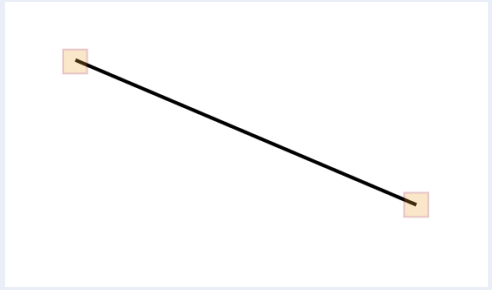


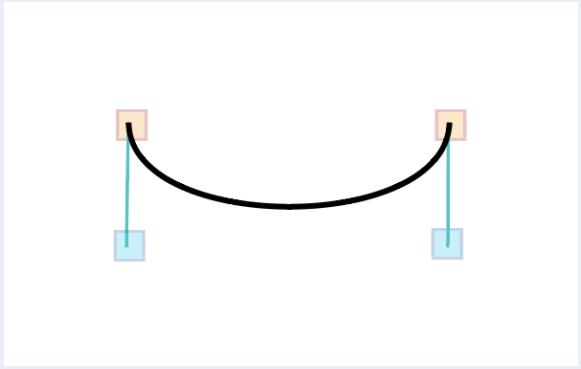
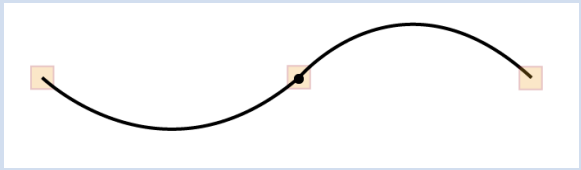
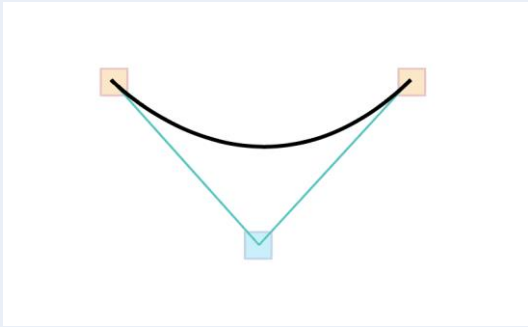
La prima cosa da notare è la rimozione della definizioni di *width* e *height*. Questi si possono dichiarare all'interno del codice CSS rendendo l'immagine molto malleabile soprattutto per lo sviluppo *responsive*

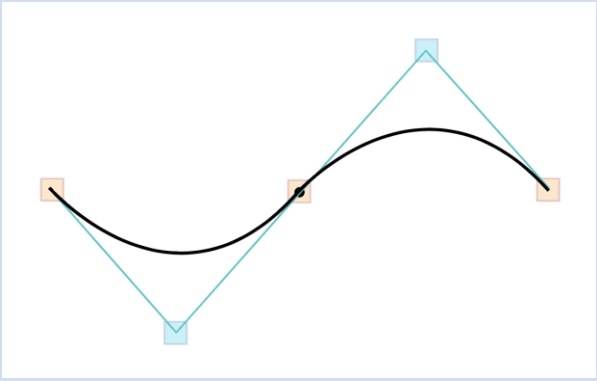
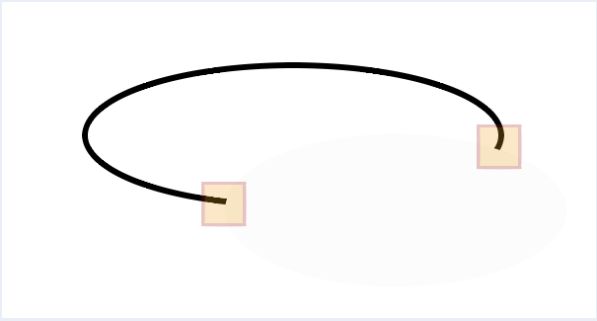
La seconda cosa è la comparsa dell'elemento «g». **g** sta per gruppo e rappresenta un modo per *nidificare e assemblare* diversi elementi insieme.

La terza cosa è la sintassi «path» che rappresenta un percorso. **path** inizia sempre con **d** (data) e contiene diversi tipi di **comandi**

Sintassi PATH

Lettera	Significato	Immagine
M, m	moveTo	Inizio del path
L, l	lineTo	
H, h	horizontal	
V, v	Vertical	

Z, z	Unisci la fine del path con il moveTo più vicino	Fine del path
C, c	Cubic Bézier	
S, s	Reflecting cubic Bézier	
Q, q	Quadratic Bézier	

T, t	Quadratic Bézier multiple	
A, a	Arco ellittico	

L’attributo viewBox

L’attributo viewBox consente al blocco SVG di essere realmente infinito e di poter controllare contemporaneamente l’area visualizzabile. Un esempio del suo utilizzo può essere trovato qui

FORME SVG

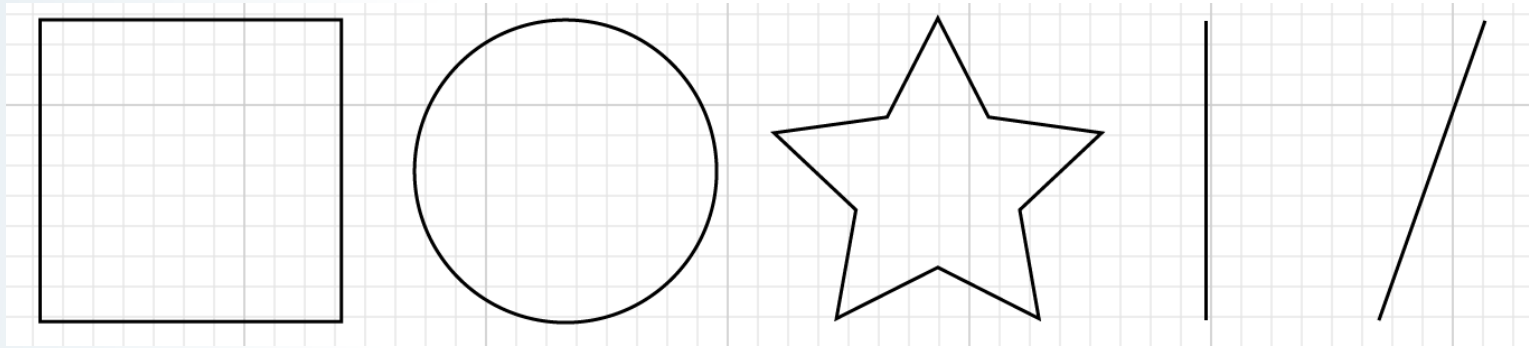
```
<rect x="10" y="5" fill="white" stroke="black" width="90" height="90"/>

<circle fill="white" stroke="black" cx="170" cy="50" r="45"/>

<polygon fill="white" stroke="black" points="279,5 294,35 328,40 303,62 309,94 279,79 248,94 254,62 230,39 263,35"/>

<line fill="none" stroke="black" x1="410" y1="95" x2="440" y2="6"/>

<line fill="none" stroke="black" x1="360" y1="6" x2="360" y2="95"/>
```



development innovation

GreenSock

GreenSock è una libreria scritta in Javascript per le animazioni WEB.

La licenza free è VERAMENTE permissiva anche a fini commerciali: finché non vendete il codice o le funzionalità di esso a più clienti, non dovete pagare nulla. Sottoscrivendo un abbonamento è possibile avere plugin aggiuntivi per semplificarsi ancor di più la vita

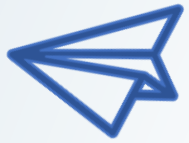
Perché usare GSAP (GreenSock Animation Platform)? La risposta è qui

1. Le animazioni create con GSAP sono 20 volte più veloci rispetto a quelle create con Jquery
2. La gestione degli easing sono già comprese nel file “TweenMax.min.js” di Greensock, basterà quindi includere documento un solo file js.
3. E’ Cross-browser, compatibile fino a IE6!
4. Con GSAP è possibile animare qualsiasi cosa all’interno di un HTML.
5. E’ possibile animare un oggetto seguendo curve di Bezier.
6. Settare un custom FPS (di default è 60fps).
7. Tutte le animazioni eseguite utilizzando GASP sono perfettamente sincronizzate.

development innovation

Sintassi base di GreenSock

```
TweenMax.to(".element", 2, { x: 100 });
```



```
TweenMax.from(".element", 2, { x: 100 });
```

```
TweenMax.fromTo(".element", 2,
                { x: 50, y: -50},
                { x: 100}
);
```

Attributi aggiuntivi

delay: è un attributo opzionale stabilisce dopo quanto dovrà iniziare l'animazione

Es: TweenMax.to(".element", 2, { x: 100, delay: 2});

ease: è un attributo opzionale, forse il più potente di Greensock, che rappresenta la velocità delle accelerazioni di un'animazione, rendendola più realistica. La maggior parte delle curve di easing hanno 3 parametri: easeIn, easeOut, easeInOut.

Uno strumento molto utile è il GreenSock Ease Visualizer (https://greensock.com/ease-visualizer)

LA TIMELINE DI GREENSOCK

Finora abbiamo usato TweenMax per realizzare le nostre animazioni. TweenMax potrebbe essere usato per tutto però viene *usato principalmente* per tutte quelle *azioni eseguite una volta sola*. Per avere animazioni eseguite in successione andrebbero inseriti numerosi delay. Lo strumento che GreenSock mette a disposizione per tali scopi è TimelineMax.

Si passa quindi da questo

```
TweenMax.to(".star", 3, {x: 300, ease: Power4.easeOut});
TweenMax.to(".circle", 3, {x: 300, delay:3, ease: Power4.easeOut});
TweenMax.to(".hex", 3, {x: 300, delay:6, ease: Power4.easeOut});
```

...

... a questo

```
var tl = new TimelineMax();

tl.to(".star", 3, {x: 300, ease: Power4.easeOut});
tl.to(".circle", 3, {x: 300, ease: Power4.easeOut});
tl.to(".hex", 3, {x: 300, ease: Power4.easeOut});
```

Il risultato si può vedere a questo link

Domanda: Cosa dovremmo fare se volessimo far partire il cerchio un secondo prima della conclusione dell'animazione della stella? (o rispetto a quando dovrebbe normalmente partire)

Risposta: usare un incremento relativo

Es: tl.to(".circle", 3, {x: 300}, "-=1");

Domanda: Se invece volessimo fare partire il cerchio dopo *n* secondi dall'inizio dell'animazione?

Risposta: usare un incremento statico

Es: tl.to(".circle", 3, {x: 300}, "2");

Domanda: Se avessimo un'animazione *molto* complessa e ci fosse la necessità di animare diversi oggetti contemporaneamente?

Risposta: Etichette relative

Etichette relative

Le etichette relative sono molto utili e potenti poiché possono essere inserite in punto qualsiasi della timeline oppure all'inizio riuscendo quindi ad animare tutti gli oggetti insieme.

```
var tl = new TimelineMax();

tl.to(".star", 3, {
  rotation: 30,
  transformOrigin: "50% 50%"
});

tl.add("burst");

tl.to(".circle1", 1, {
  scale: 2.5
  x: 100,
  y: -70
}, "burst");

tl.to(".circle2", 1, {
  scale: 2.5
  y: -100
}, "burst");
```

Con questo approccio la leggibilità e la comprensione aumentano. È un sistema anche molto flessibile: anche se ci dovessero essere delle modifiche temporali nella timeline (in anticipo o in ritardo) gli elementi legati all'etichetta si muoveranno comunque insieme.

Esempio

Anche con le etichette relative è possibile usare un incremento relativo.

Es: tl.to('.circle', 1, {x: 100}, "burst+=1");

ALTRI METODI E ATTRIBUTI DI GREENSOCK

Metodo set: serve per impostare uno o più parametri di un oggetto prima che l'animazione di questo inizi.

Es: tl.set(".circle", {scale: 0.5});

Metodo seek: fa vedere ciò che succedere da un certo istante in poi.

Es: tl.seek('inizio+=2');

Metodo timescale(sec): aumenta (valori > 1) o diminuisce (valori < 1) la velocità dell'animazione.

Es: tl.timeScale(1.5);

Attributo scale: ingrandisce (valori > 1) o diminuisce (valori < 1) l'oggetto relativo.

Es: tl.to(".circle", 2, {scale: 3});

Attributo repeat: ripete un'animazione *n* volte in più. Può valere 0 (default), numeri >=1 o -1(infinito).

Es: tl.to(".circle", 2, {repeat: 1});

Attributo yoyo: consente all'animazione di ''oscillare avanti e indietro''. Può assumere come valori *true* o *false* e viene applicata solo se *repeat* è definito.

Es: tl.to(".circle", 2, {repeat: -1, yoyo: true});

Attributo rotation: imposta l'angolo di rotazione. Usato spesso con l'attributo transformOrigin (opzionale con valore default '0 0') che indica il punto di origine della rotazione. transformOrigin può essere espresso in % o con top/bottom/left/right/center. L'origine è sempre l'angolo left/top.

Es: tl.to(".circle", 2, {rotation: 90, transformOrigin: '0 0'});

ALCUNE POSSIBILI APPLICAZIONI

1. Fun 1
2. Fun 2
3. Infografica
4. Diagramma di flusso