EBook Gratuito

APPRENDIMENTO SVG

Free unaffiliated eBook created from
Stack Overflow contributors.

#svg

Sommario

Di

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: svg

It is an unofficial and free SVG ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official SVG.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capitolo 1: Iniziare con SVG

Osservazioni

Scalable Vector Graphics (SVG) è uno standard W3C per il disegno di immagini vettoriali.

Ecco un semplice file SVG standalone:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <circle cx="50" cy="50" r="25" fill="blue"/>
</svg>
```

SVG può anche essere incorporato in HTML, nel qual caso l'attributo `xmlns` non è richiesto.

Altri elementi grafici sono:

- `<line>`
- `<ellipse>`
- `<path>`
- `<polygon>` e `<polyline>`
- `<text>` inclusi elementi `<tspan>` come `<tspan>` e `<textPath>`

Il CSS è usato per lo styling sebbene non tutte le proprietà CSS si applichino a SVG e SVG stesso definisce alcune proprietà specifiche come il `fill` e il `stroke` che non sono usati altrove.

Le forme possono essere riempite con gradienti o motivi e si possono ottenere effetti raster aggiuntivi usando i filtri.

Il ritaglio è disponibile utilizzando gli elementi grafici sopra indicati come percorsi di clip.

Riguardo alle versioni dello standard SVG W3C:

- La versione corrente è SVG 1.1 (Seconda edizione)
- Il W3C sta attualmente lavorando su una bozza di SVG 2

Versioni

Versione	Data di rilascio
1.0	2001/09/04
1.1 Prima edizione	2003/01/14
1.2 Piccolo	2008-12-22
1.1 Seconda edizione	2011-08-16

Examples

Inline SVG

Inline SVG consente al markup SVG, scritto in HTML, di generare grafica nel browser.

Quando si utilizza SVG in linea, un DOCTYPE non è strettamente richiesto. Invece, solo i `<svg>` apertura e chiusura insieme a un viewBox oa attributi di larghezza e altezza saranno sufficienti:

```
<svg width="100%" height="100%">
    <!-- SVG elements go here -->
</svg>
```

Il frammento `<svg>` sopra agisce sia come contenitore sia come elemento strutturale. Questo frammento stabilisce il proprio sistema di coordinate.

Di seguito è riportato un esempio di rendering di un frammento SVG con alcuni contenuti. Produrrà un rettangolo con "Hello World!" testo al suo interno.

```
<svg width="50%" viewBox="0 0 10 10">
    <rect x="1" y="1" width="5" height="3" fill="teal" />
    <text x="2" y="2" font-family="Palatino, Georgia, serif" font-size="3%" font-weight="bold"
fill="white">Hello World!</text>
</svg>
```

Risultato:

SVG come

È possibile eseguire il rendering dei contenuti di un file SVG come immagine all'interno di un documento HTML utilizzando un tag `<img>` . Per esempio:

```
<img src="my_svg_file.svg" alt="Image description">
```

Per impostazione predefinita, le dimensioni dell'immagine verranno visualizzate in base alle proprietà **width** e **height** specificate nel file SVG a cui si fa riferimento nell'attributo `src` .

Vale la pena notare le varie limitazioni inerenti a questo approccio:

- Il supporto del browser, sebbene buono, non include Internet Explorer 8 e versioni precedenti, né Android 2.3 e versioni precedenti.
- Non è possibile assegnare uno stile ai singoli elementi contenuti nel file SVG usando CSS che è esterno al file SVG. Tutto il CSS deve essere all'interno del file immagine stesso.
- JavaScript non verrà eseguito.
- L'immagine deve essere completa in un singolo file. Ad esempio, se il file SVG contiene immagini raster, quelle immagini interne devono essere codificate come URL di dati.

SVG come immagine di sfondo

È possibile visualizzare un file SVG all'interno di un documento HTML, specificandolo come immagine di sfondo in CSS. Per esempio:

```
.element {
    background-size: 100px 100px;
    background: url(my_svg_file.svg);
    height: 100px;
    width: 100px;
}
```

Se le dimensioni specificate nel file SVG sono maggiori delle dimensioni dell'elemento HTML, potrebbe essere opportuno specificare la proprietà `background-size` per ridimensionare l'SVG in modo che si adatti al suo elemento.

Come con l'utilizzo di SVG come `<img>` , vale la pena notare alcune limitazioni con questo approccio:

- Il supporto del browser non include Internet Explorer 8 e versioni precedenti, né Android 2.3 e versioni precedenti.
- Non è possibile assegnare uno stile ai singoli elementi contenuti nel file SVG usando CSS che è esterno al file SVG. Tutto il CSS deve essere all'interno del file immagine stesso.

Leggi Iniziare con SVG online: https://riptutorial.com/it/svg/topic/963/iniziare-con-svg

Capitolo 2: Animazione

Osservazioni

L'animazione SMIL tramite l'elemento `<animate>` è attualmente (luglio 2016) supportata nei principali browser ad eccezione dei browser Microsoft. Esiste una libreria (fakeSMIL) che può essere utilizzata come polyfill per la compatibilità Microsoft.

Chrome 45 ha perso SMIL in favore delle animazioni CSS e della prossima sintassi di animazione dichiarativa delle animazioni Web, che purtroppo è solo parzialmente implementata nei browser correnti. Ma gli sviluppatori di Chrome hanno recentemente sospeso il loro intento (vedi questa risposta StackOverflow)

Examples

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="50" y="50" height="100" width="100" stroke="black" fill="yellow">
        <animate
            attributeType="XML"
            attributeName="height"
            begin="0s"
            dur="10s"
            from="100"
            to="200"
            repeatCount="indefinite"
        />
    </rect>
</svg>
```

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="50" y="50" height="100" width="100" stroke="black" fill="yellow">
        <animateTransform
            attributeType="XML"
            attributeName="transform"
            type="rotate"
            begin="0s"
            dur="10s"
            from="0"
            to="360"
            repeatCount="indefinite"
        />
    </rect>
</svg>
```

Leggi Animazione online: https://riptutorial.com/it/svg/topic/3260/animazione

Capitolo 3: Cerchio

Parametri

parametri	Dettagli
cx	coordinata x del centro del cerchio.
cy	coordinata y del centro del cerchio.
r	Raggio di cerchio.
ictus	Colore del bordo del cerchio.
riempire	Colore *all'interno del* bordo del cerchio.

Osservazioni

Informazioni dettagliate sull'elemento 'cerchio' SVG sono disponibili nella Raccomandazione W3C per SVG .

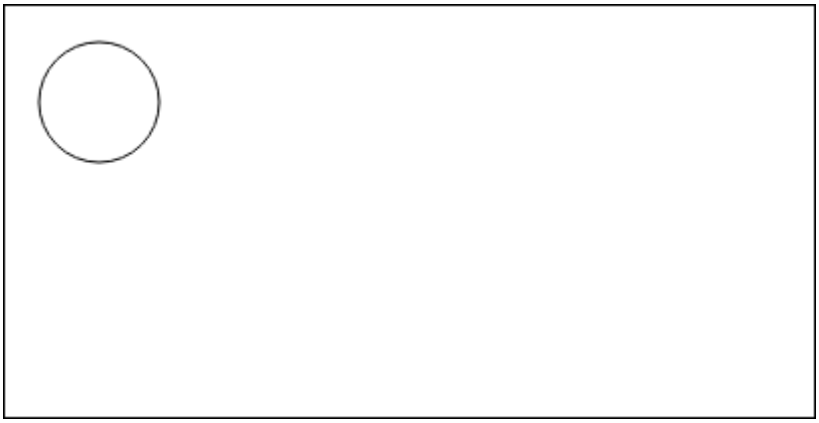
Examples

Disegna un cerchio nero senza riempimento

- I valori `cx` e `cy` indicano la posizione del centro del cerchio.
- L'attributo `r` specifica la dimensione del raggio del cerchio.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <circle cx="40" cy="40" r="30" stroke="black" fill="none" />
</svg>
```

Risultato:



Leggi Cerchio online: https://riptutorial.com/it/svg/topic/1559/cerchio

Capitolo 4: clipPath

Parametri

Parametro	Descrizione
clipPathUnits	il sistema di coordinate dei contenuti del pattern è *objectBoundingBox* o *userSpaceOnUse*

Osservazioni

Informazioni sulla raccomandazione relative al W3C

Examples

Ritaglio su un percorso circolare

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 100"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <defs>
    <clipPath id="circleClip">
      <circle cx="50" cy="60" r="20" />
    </clipPath>
  </defs>
  <image width="100" height="100" style="clip-path:url(#circleClip)"
xlink:href="https://cdn.sstatic.net/Sites/stackoverflow/company/img/logos/so/so-icon.png" />
</svg>
```

Logo originale	Logo dopo il ritaglio
	

Leggi clipPath online: https://riptutorial.com/it/svg/topic/4840/clippath

Capitolo 5: Colori

Examples

Colori con nome: utilizzare nomi predefiniti per gli attributi di riempimento e traccia

Un elenco di nomi di parole chiave a colori riconosciuti può essere trovato nella Raccomandazione W3C per SVG .

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="red" stroke="green" />
  <rect x="200" y="200" width="50" height="50" fill="yellow" stroke="blue" />
</svg>
```

Colori RGB usando la notazione esadecimale

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="#ff0000" stroke="#00ff00" />
  <rect x="200" y="200" width="50" height="50" fill="#ffff00" stroke="#00ffff" />
</svg>
```

Come sopra usando la forma esadecimale di stenografia :

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="#f00" stroke="#0f0" />
  <rect x="200" y="200" width="50" height="50" fill="#ff0" stroke="#0ff" />
</svg>
```

Colori RGB con notazione funzionale - valori interi o percentuali

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="rgb(255, 0, 0)" stroke="rgb(0, 255, 0)" />
  <rect x="200" y="200" width="50" height="50" fill="rgb(100%, 100%, 0%)" stroke="rgb(0%,
100%, 100%)" />
</svg>
```

nella notazione funzionale, sono supportati anche i valori RGBA.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <circle r="30" cx="100" cy="100" fill="rgba(255, 0, 0, 0.5)" stroke="rgba(0, 255, 0,5
0.5)" />
    <rect x="200" y="200" width="50" height="50" fill="rgba(100%, 100%, 0%, 0.5)"
stroke="rgba(0, 100%, 100%, 0.5)" />
</svg>
```

La parola chiave currentColor

`currentColor` è più utile negli SVG in linea. Con questo puoi ereditare il colore CSS dei genitori e usarlo ovunque i colori sono usati in SVG.

In questo esempio, il primo cerchio utilizza il colore del testo come colore di riempimento e il secondo cerchio lo utilizza come colore del tratto.

```
<html>
    <head>
        div{color:green}
    </head>
    <body>
        <div>
            some Text
            <svg width="2em" height="1em" viewBox="0 0 200 100">
                <circle cx="50" cy="50" r="45" fill="currentColor"/>
                <circle cx="150" cy="50" r="45" fill="none" stroke-width=5
stroke="currentColor"/>
            </svg>
        </div>
    </body>
</html>
```

Leggi Colori online: https://riptutorial.com/it/svg/topic/2463/colori

Capitolo 6: Creazione di caratteri

introduzione

I font SVG non sono più supportati direttamente dai browser. Tuttavia sono molto utili per generare a livello di programmazione caratteri come caratteri simbolo o caratteri di codici a barre. Ci sono molti strumenti che ti permettono di convertire i font svg in qualsiasi altro formato di font.

Osservazioni

Ecco una lista di strumenti che puoi usare con i font SVG.

Convertitori

- https://github.com/fontello/svg2ttf

Examples

un semplice font

Un semplice esempio di un font svg. Alcune cose da notare qui:

- il sistema di coordinate dei glifi è in opposizione al solito sistema di coordinate in svg. **l'asse y è rivolto verso l'alto** . Il punto 0,0 è nell'angolo in basso a destra.
- Tutti i percorsi di un font devono essere disegnati in senso antiorario.
- Nella maggior parte degli strumenti è supportato solo l'attributo d dell'elemento glifo. Gli elementi secondari non funzioneranno, sebbene siano tecnicamente consentiti.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <font id = "myFont"
        horiz-adv-x   = "1000"
        vert-origin-x = "0"
        vert-origin-y = "0" >
    <font-face font-family  = "myFont"
               font-weight  = "normal"
               units-per-em = "1000">
      <font-face-src>
        <font-face-name name="myFont"/>
      </font-face-src>
    </font-face>`
    <glyph unicode="a" d="M0 0 H1000 L500 1000z M200 200 L500 800 L800 200z" />
    <glyph unicode="b" d="M0 0 H1000 L500 1000z M200 200 L500 800 L800 200z" />
  </font>
</svg>
```

Se hai glifi più larghi o più stretti, modifica l'horiz-adv-x sull'elemento glifo stesso.

```
<glyph unicode="a" horiz-adv-x="512" d="M0 0 H1000 L500 1000z M200 200 L500 800 L800 200z" />
```

raccolta di caratteri

la proprietà unicode viene utilizzata per la selezione di glifi successiva. È possibile utilizzare lettere semplici o codici unicode e legature (combinazione di lettere o codepoint unicode)

- `unicode="abc"`
- `unicode="&#97;&#98;"`
- `unicode="ab&#97;&#98;"`
- `unicode="a"`
- `unicode="&#98;"`

i glifi vengono sempre selezionati per prima corrispondenza, quindi avere tutte le legature prima di ogni singolo carattere.

i codepoint di unicode possono essere scritti in decimale `&#123;` o in notazione esadecimale `&#x1f` .

salita, discesa e linea di base

la proprietà units-per-em è una delle proprietà dei font più importanti. È usato per dare qualsiasi valore a qualsiasi altra proprietà.

la specifica del font CSS2 ha una bella definizione di em sqare :

> Alcuni valori, come le metriche di larghezza, sono espressi in unità relative a un quadrato astratto la cui altezza è la distanza prevista tra le righe di tipo nella stessa dimensione del tipo

la **linea di base predefinita è a 0** nel quadrato em. per il calcolo dell'altezza della linea e per gli scopi di allineamento, le due ascensioni e discese delle preriffe sono di estrema importanza.

L'ascesa è la distanza massima dalla linea di base al punto più alto del tuo glifo più grande. Usualmente questo è 1em, quindi il valore che hai dato per unità-per-em.

La discesa è la distanza massima dalla linea di base al punto più basso in qualsiasi glifo del tuo carattere.

Ecco un font con glifi che mostra una linea nel punto più basso e più alto così come nella linea di base.

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1000 1000">
  <font id = "myFont"
        horiz-adv-x   = "1000"
        vert-origin-x = "0"
        vert-origin-y = "0" >
    <font-face font-family  = "myFont"
               font-weight  = "normal"
               units-per-em = "1000"
               descent="500"
               ascent="1000">
```

```
      <font-face-src>
        <font-face-name name="myFont"/>
      </font-face-src>
    </font-face>`
    <glyph unicode = "a" d = "M0 900h1000v100h-1000z" />
    <glyph unicode = "b" d = "M0 0h1000v100h-1000z" />
    <glyph unicode = "c" d = "M0 -500h1000v100h-1000z" />
  </font>
</svg>
```

Ascesa e discesa sono utilizzate per determinare l'altezza della linea. Unità-per-em e baseline sono utilizzate per determinare la posizione e le dimensioni relative ad altri font utilizzati ..

Leggi Creazione di caratteri online: https://riptutorial.com/it/svg/topic/8147/creazione-di-caratteri

Capitolo 7: defs

Sintassi

- `<defs>` ... *elementi definiti* ... `</defs>`

Parametri

Parametro	Dettagli
defs	L'elemento defs non ha parametri

Osservazioni

L'elemento `<defs>` viene utilizzato come elemento contenitore per elementi che sono destinati ad essere utilizzati esclusivamente per riferimento e non resi direttamente. Gli elementi che verrebbero normalmente renderizzati (ad esempio `<rect>` , `<circle>`) dichiarati all'interno di un blocco `<defs>` vengono trattati come se il loro stile includesse la `display:none` .

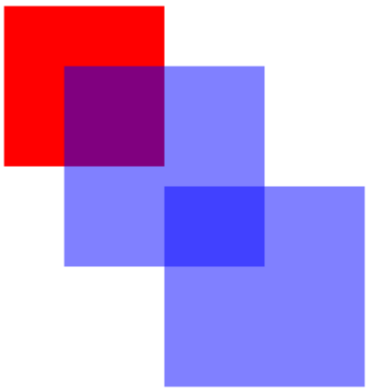
Anche se non è strettamente necessario, le specifiche SVG. raccomanda di inserire tutte le definizioni di gradiente, filtro, modello, maschera, simbolo e marcatore all'interno di un blocco di `defs` .

Examples

Di base esempio

```
<svg width="400px" height="400px">
<defs>
  <rect id="defrect" fill="blue" fill-opacity=".5" x="50" y="50" width="100" height="100"/>
</defs>

<rect fill="red" x="20" y="20" width="80" height="80"/>
<use xlink:href="#defrect"/>
<use xlink:href="#defrect" x="50" y="60"/>

</svg>
```

Risultato



Leggi defs online: https://riptutorial.com/it/svg/topic/5592/defs

Capitolo 8: Ellisse

Parametri

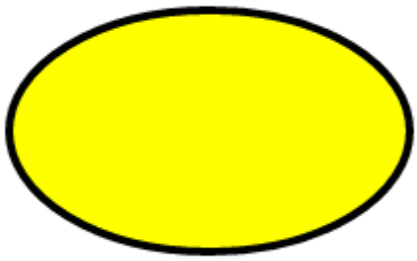
Parametro	Dettagli
`cx`	Coordinata X del centro dell'ellisse
`cy`	Coordinata Y del centro dell'ellisse
`rx`	Raggio orizzontale
`ry`	Raggio verticale

Examples

Semplice ellisse gialla

```
<svg height="80" width="160">
  <ellipse cx="80" cy="40" rx="50" ry="30"
    style="fill:yellow; stroke:black; stroke-width:2" />
</svg>
```

resa:



Leggi Ellisse online: https://riptutorial.com/it/svg/topic/3993/ellisse

Capitolo 9: filtri

Sintassi

- Dichiarazione filtro: `<filter id="filter-id"` > ... elenco di primitive figlio ... `</filter>`
- Applica il filtro tramite l'attributo SVG: `<elementname filter="url(#filter-id)" ... />`
- Applica il filtro tramite la proprietà CSS: *(-prefix-*) filter: url ("# filter-id");

Parametri

Attributi elemento	Dettagli
Regione del filtro	L'elemento filtro può facoltativamente definire la posizione, le dimensioni, la risoluzione e le unità per l'uscita di un effetto filtro. La posizione e le dimensioni di un filtro possono essere specificate utilizzando i seguenti parametri: x, y, larghezza, altezza. I valori predefiniti *non* sono *intuitivi* e sono: x: -10% y: -10% larghezza: 120% altezza: 120%
Filtro di risoluzione	L'attributo `filterRes` è un attributo facoltativo in SVG 1.1 che può essere utilizzato per specificare la risoluzione con cui viene elaborato il filtro. Questo attributo aveva un supporto non uniforme e ora è deprecato nei browser correnti.
Unità di filtro	Per impostazione predefinita, le unità e il sistema di coordinate per la regione degli effetti del filtro (x, y, larghezza, altezza) di un elemento del filtro sono impostati rispetto al riquadro di delimitazione dell'elemento che fa riferimento al filtro. Nei termini SVG, questo è chiamato "objectBoundingBox". Quando scriviamo x = "50%" significa "imposta la posizione x iniziale della regione del filtro sul lato sinistro del riquadro di delimitazione dell'elemento di riferimento + 50% della larghezza dell'elemento". Ma puoi anche specificare le unità e le coordinate esplicitamente impostando la proprietà filterUnits. Le due alternative sono "objectBoundingBox" (l'impostazione predefinita che abbiamo appena descritto) o "userSpaceOnUse". userSpaceOnUse imposta le unità filtro e il sistema di coordinate sull'area di disegno dell'elemento di riferimento, o in termini SVG, "userSpaceOnUse".
Unità primitive	Oltre al sistema di unità per il filtro stesso, è possibile specificare anche il sistema di unità che verrà utilizzato dalle primitive del filtro figlio del filtro tramite l'attributo primitiveUnits. Ancora una volta, la scelta è tra userSpaceOnUse e objectBoundingBox. Questi influenzano le coordinate 0,0 e i valori unitari per i primitivi del filtro nello stesso modo di filterUnits.
Spazio colore	Lo spazio colore predefinito per i filtri SVG è linearRGB. L'attributo opzionale `color-interpolation-filters` può essere impostato su `sRGB` per cambiare lo

Attributi elemento	Dettagli
	spazio colore allo spazio sRGB più convenzionale.

Osservazioni

La maggior parte degli attributi del filtro sono animabili tramite l'elemento `<animate>` , sebbene sia necessario utilizzare la libreria "fakeSMIL" su IE per ottenere gli stessi risultati. L'animazione SMIL (l'elemento `<animate>`) sarà deprecata a favore della nuova specifica Web Animations, che ha un supporto molto limitato a partire dalla metà del 2016.

Elementi secondari dell'elemento Filtro - primitive del filtro - hanno due attributi facoltativi che specificano lo spazio colore all'interno del quale vengono eseguiti i calcoli di interpolazione del colore: interpolazione del colore e filtri di interpolazione del colore. L'impostazione predefinita per la prima è sRGB e l'impostazione predefinita per quest'ultima è linearRGB. Manipolazioni che invertono lo spazio cromatico (tramite feColorMatrix o feComponentTransfer) possono risultare in risultati non intuitivi, per quelli utilizzati nello spazio colore CSS sRGB. Ad esempio, l'inversione del colore di un'immagine in scala di grigi in linearRGB provocherà uno spostamento pronunciato verso i toni più bianchi. Questo può essere corretto impostando il valore della primitiva su sRGB. Questi attributi possono essere impostati sulle singole primitive del filtro o ereditati dall'elemento filtro stesso.

Se non viene specificato nessun altro input, ma ne è richiesto uno, la prima primitiva del filtro all'interno di un filtro assumerà come input la versione rasterizzata (bitmap) dell'elemento di riferimento. Le successive primitive di filtro che prevedono un input prenderanno come input il risultato della primitiva del filtro immediatamente precedente.

Nei filtri complessi, può diventare difficile tenere traccia degli input e degli output (e debuggarli) se sono lasciati impliciti; ed è buona pratica dichiarare esplicitamente input e output per ogni primitiva.

I primitivi del filtro SVG possono essere suddivisi colloquialmente in input, trasformazioni, effetti di illuminazione e combinazioni.

ingressi:

feFlood: genera un campo di colore

feTurbulence: genera un'ampia varietà di effetti di rumore

feImage: genera un'immagine da un riferimento di immagine esterna, URI di dati o riferimento a un oggetto (i riferimenti a oggetti non sono supportati in Firefox a partire da metà dicembre '12)

trasformazioni:

feColorMatrix: trasforma i valori di input di un pixel RBGA in valori di output

feComponentTransfer: regola la curva del colore di un singolo canale di colore

feConvolveMatrix: sostituisce ogni pixel con un nuovo pixel calcolato dai valori dei pixel in un'area relativa al pixel corrente)

feGaussianBlur: sostituisce il pixel corrente con una media ponderata di pixel in un'area attorno al pixel

feDisplacementMap: sposta ciascun pixel dalla sua posizione corrente in base ai valori R, G o B da un altro grafico di input.

feMorfologia: sostituisce ogni pixel con un nuovo pixel calcolato dal valore massimo o minimo di tutti i pixel in un'area rettangolare attorno a quel pixel.

feOffset: sposta l'input dalla sua posizione corrente

Effetti luminosi:

feSpecularLighting: fornisce un effetto di illuminazione "brillante" 2D o pseudo-3D

feDiffuseLighting: fornisce un effetto di illuminazione "opaca" 2D o pseudo-3D

feDistantLight: fornisce una fonte di luce distante per l'illuminazione speculare o diffusa

feSpotLight: fornisce una sorgente di luce a sezione conica per illuminazione speculare o diffusa

fePointLight: fornisce una sorgente di luce puntiforme per illuminazione speculare o diffusa

combinazioni:

feMerge: crea un semplice over-composito da più input (inclusi i precedenti input dei filtri)

feBlend: miscela più input usando le regole di mixaggio

feComposite: combina più input utilizzando le regole di combinazione impostate, tenendo conto dei valori alfa.

feTile: tessere immesse per creare un motivo ripetuto

Altre note

Sebbene SVG sia una tecnologia di grafica vettoriale, è importante sottolineare che i filtri SVG eseguono operazioni a *livello di pixel* su tutti gli input (incluse le forme SVG) e producono output rasterizzati (bitmap) con un livello di risoluzione specificato. L'applicazione di una trasformazione in scala 10x (ad esempio) su una curva SVG semplice che è stata filtrata a una normale risoluzione dello schermo produrrà bordi pixelati poiché l'anti-aliasing dell'elemento grafico originale è stato convertito in pixel dal filtro e ingrandito. (Non è chiaro se questo è conforme alle specifiche o solo una limitazione delle attuali implementazioni)

Ricorda che SVG è XML quando scrivi filtri, quindi tutti i tag devono essere chiusi e molte

proprietà e attributi devono essere specificati esplicitamente o il filtro non verrà eseguito.

Un elemento filtro non viene mai visualizzato direttamente. Viene fatto riferimento solo utilizzando la proprietà filter sull'elemento a cui viene applicato il filtro. Si noti che la proprietà display non si applica all'elemento filtro e gli elementi non vengono visualizzati direttamente anche se la proprietà display è impostata su un valore diverso da "none". Al contrario, gli elementi filtro sono disponibili per il riferimento anche quando la proprietà di visualizzazione sul filtro o su uno qualsiasi dei suoi antenati è impostata su "nessuno".

I filtri SVG possono essere referenziati tramite un filtro CSS, anche se a metà 2016, solo un sottoinsieme di primitive è supportato tramite questo meccanismo e questo meccanismo non è supportato nei browser Microsoft.

Examples

Blur Filters: feGaussian Blur (basic)

```
<svg width="900px" height="400px" viewBox="0 0 900 400">
   <defs>
      <filter id="basicGaussian">
         <feGaussianBlur stdDeviation="5"/>
      </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#basicGaussian)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

(Immagine sorgente di *Daderot* su Wikimedia Commons)

Filtri sfocatura: feGaussianBlur (set di sfocatura dell'asse x e dell'asse y separatamente)

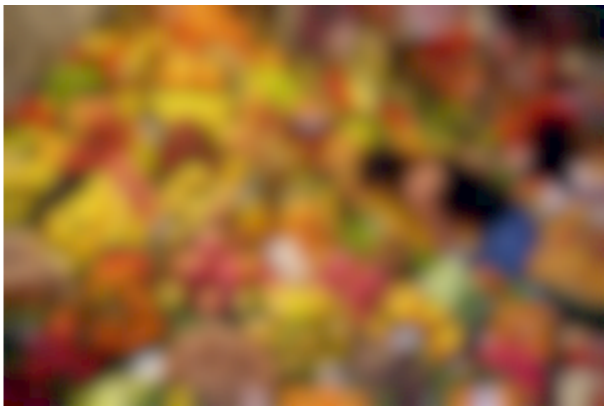
```
<svg width="900px" height="400px" viewBox="0 0 900 400">
   <defs>
```

```
      <filter id="xAxisGaussian">
         <feGaussianBlur stdDeviation="5 0"/>
      </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#xAxisGaussian)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

(Immagine sorgente di *Daderot* su Wikimedia Commons)

Filtri di sfocatura: feGaussianBlur con bordi rigidi e 100% di opacità

```
<svg width="900px" height="400px" viewBox="900 400">
   <defs>
      <filter id="GaussianHardEdge" x="0%" y="0%" width="100%" height="100%">
         <feGaussianBlur stdDeviation="5"/>
         <feComponentTransfer>
           <feFuncA type="table" tableValues="1 1"/>
         </feComponentTransfer>
      </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#GaussianHardEdge)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

(Immagine sorgente di *Daderot* su Wikimedia Commons)

Filtri sfocatura: Box Blur

```
<svg width="900px" height="400px" viewBox="900 400">
   <defs>
      <filter id="GaussianHardEdge" >
         <feConvolveMatrix order="3" kernelMatrix=" 1 1 1
                                                    1 1 1
                                                    1 1 1"/>
      </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#GaussianHardEdge)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Immagine sorgente di *Daderot* su Wikimedia Commons)

Filtri sfocatura: Bokeh Blur (3 strati, ritagliato)

```
<svg width="900px" height="400px" viewBox="0 0 900 400">
   <defs>
<filter id="BokehBlur" color-interpolation-filters="sRGB">
      <feGaussianBlur stdDeviation="2" result="blurSource"/>
      <feColorMatrix type="luminanceToAlpha"/>
      <feComponentTransfer result="brightness-mask" >
        <feFuncA type="discrete" tableValues="0 0 0 1 1"/>
```

```
    </feComponentTransfer>


 <!--bokeh Layer 1 -->
 <feTurbulence type="fractalNoise" seed="1" baseFrequency=".67" numOctaves="3"/>
<feColorMatrix type="luminanceToAlpha"/>
  <feComponentTransfer>
    <feFuncA type="discrete" tableValues="0 0 0 1"/>
 </feComponentTransfer>
  <feComposite operator="in" in="brightness-mask"/>
  <feComposite operator="in" in="blurSource"/>

  <feMorphology operator="dilate" radius="5"/>
  <feGaussianBlur stdDeviation="8"/>
  <feColorMatrix type="matrix" values="1 0 0 0 0  0 1 0 0 0  0 0 1 0 0
                                       0 0 0 9 0" />
  <feComponentTransfer result="bokeh1">
    <feFuncA type="linear" slope=".5" />
 </feComponentTransfer>



  <!--bokeh Layer 2 -->
 <feTurbulence type="fractalNoise" seed="49" baseFrequency=".67" numOctaves="3"/>
<feColorMatrix type="luminanceToAlpha"/>
  <feComponentTransfer>
    <feFuncA type="discrete" tableValues="0 0 0 1"/>
 </feComponentTransfer>
  <feComposite operator="in" in="brightness-mask"/>
  <feComposite operator="in" in="blurSource"/>

  <feMorphology operator="dilate" radius="10"/>
  <feGaussianBlur stdDeviation="12"/>
  <feColorMatrix type="matrix" values="1 0 0 0 0  0 1 0 0 0  0 0 1 0 0
                                       0 0 0 15 0" />
  <feComponentTransfer result="bokeh2">
    <feFuncA type="linear" slope=".3" />
 </feComponentTransfer>

<!--bokeh Layer 3 -->

<feTurbulence type="fractalNoise" seed="44" baseFrequency=".67" numOctaves="3"/>
<feColorMatrix type="luminanceToAlpha"/>
  <feComponentTransfer>
    <feFuncA type="discrete" tableValues="0 0 0 1"/>
 </feComponentTransfer>
  <feComposite operator="in" in="brightness-mask"/>
  <feComposite operator="in" in="blurSource"/>

  <feMorphology operator="dilate" radius="10"/>
  <feGaussianBlur stdDeviation="18"/>
  <feColorMatrix type="matrix" values="1 0 0 0 0  0 1 0 0 0  0 0 1 0 0
                                       0 0 0 15 0" />
  <feComponentTransfer result="bokeh3">
    <feFuncA type="linear" slope=".2" />
 </feComponentTransfer>

<!--Merge -->
 <feBlend mode="multiply" in="bokeh3" in2="bokeh2"/>
 <feBlend mode="lighten" in2="bokeh1"/>
```

```
      <feMorphology operator="erode" radius="0" result="bokeh"/>
      <feGaussianBlur stdDeviation="9" in="SourceGraphic"/>
      <feComposite operator="over" in="bokeh"/>
      <feComposite operator="in" in2="SourceGraphic"/>

    </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#BokehBlur)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Immagine sorgente di *Daderot* su Wikimedia Commons)

Filtri ombra: Basic Dropshadow

```
<svg width="800px" height="600px">
<defs>
  <filter id="drop-shadow">
    <feGaussianBlur in="SourceAlpha" stdDeviation="4"/>
    <feOffset dx="5" dy="5" result="offsetblur"/>
    <feFlood flood-color="red"/>
    <feComposite in2="offsetblur" operator="in"/>
    <feMerge>
      <feMergeNode/>
      <feMergeNode in="SourceGraphic"/>
  </feMerge>
</filter>
  </defs>

  <text filter="url(#drop-shadow)" x="30" y="100" font-size="80">SVG Filters</text>

</svg>
```

Filtri ombra: bagliore interno

```
<svg width="800px" height="600px">
<defs>
  <filter id="inner-glow">
    <feFlood flood-color="red"/>
    <feComposite in2="SourceAlpha" operator="out"/>
```

```
    <feGaussianBlur stdDeviation="2" result="blur"/>
    <feComposite operator="atop" in2="SourceGraphic"/>
</filter>
  </defs>

  <text filter="url(#inner-glow)" x="30" y="100" font-size="80" font-family="Sans-Serif" font-
weight="bold">SVG Filters</text>

</svg>
```

Filtri ombra: ombretto complesso (sagomato, rumoroso, a forma di)

```
<svg width="800px" height="600px">
<defs>
<filter id="complex-shadow" color-interpolation-filters="sRGB" x="-50%" y="-50%" height="200%"
width="200%">

<!-- Take source alpha, offset it by angle/distance and blur it by size -->
<feOffset id="offset" in="SourceAlpha" dx="11" dy="6" result="SA-offset"/>
<feGaussianBlur id="blur" in="SA-offset" stdDeviation="4" result="SA-o-blur"/>

<!-- Apply a contour by using a color curve transform on the alpha and clipping the result to
the input -->

<feComponentTransfer in="SA-o-blur" result="SA-o-b-contIN">
  <feFuncA id="contour" type="table" tableValues="0 1 .3 .1 0.05 .1 .3 1 "/>
</feComponentTransfer>

<feComposite operator="in" in="SA-o-blur" in2="SA-o-b-contIN" result="SA-o-b-cont"/>

<!-- Adjust the spread by multiplying alpha by a constant factor --> <feComponentTransfer
in="SA-o-b-cont" result="SA-o-b-c-sprd">
  <feFuncA id="spread-ctrl" type="linear" slope="2.8"/>
</feComponentTransfer>

<!-- Adjust color and opacity by adding fixed offsets and an opacity multiplier -->
<feColorMatrix id="recolor" in="SA-o-b-c-sprd" type="matrix" values="0 0 0 0 0.945 0 0 0 0 0
0.137 0 0 0 0 0.137 0 0 0 0.49 0" result="SA-o-b-c-s-recolor"/>

<!-- Generate a grainy noise input with baseFrequency between approx .5 to 2.0. And add the
noise with k1 and k2 multipliers that sum to 1 -->
<feTurbulence result="fNoise" type="fractalNoise" numOctaves="6" baseFrequency="1.98"/>
<feColorMatrix in="fNoise" type="matrix" values="1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 7 -3"
result="clipNoise"/>
<feComposite id="noisemix" operator="arithmetic" in="SA-o-b-c-s-recolor" in2="clipNoise"
k1="0.67" k2="0.33" result="SA-o-b-c-s-r-mix"/>

<!-- Merge the shadow with the original -->
<feMerge>
  <feMergeNode in="SA-o-b-c-s-r-mix"/>
  <feMergeNode in="SourceGraphic"/>
</feMerge>
</filter>
  </defs>

  <text filter="url(#complex-shadow)" x="30" y="100" font-size="80" font-family="Sans-Serif"
font-weight="bold">SVG Filters</text>

</svg>
```

Filtri per la manipolazione del colore: livelli di grigio di base

```
<svg width="800px" height="600px">
   <defs>
     <filter id="greyscale">
     <feColorMatrix type="matrix"
             values="0.2126  0.7152  0.0722  0 0
                     0.2126  0.7152  0.0722  0 0
                     0.2126  0.7152  0.0722  0 0
                     0 0 0 1 0"/>
     </filter>
   </defs>

   <image

xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#greyscale)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Immagine sorgente di *Daderot* su Wikimedia Commons)

Filtri di manipolazione del colore: scala di grigi (solo canale verde)

```
<svg width="800px" height="600px">
   <defs>
     <filter id="greyscale">
     <feColorMatrix type="matrix"
             values="0  1  0  0 0
                     0  1  0  0 0
                     0  1  0  0 0
                     0 0 0 1 0"/>
     </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#greyscale)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Immagine sorgente di *Daderot* su Wikimedia Commons)

Filtri per la manipolazione del colore: monotono

```
<svg width="800px" height="600px">
   <defs>
     <filter id="greyscale">
     <feColorMatrix type="matrix"
             values=".2  .2  .2  0 0
                     .6  .6  .6  0 0
                     .2  .2  .2  0 0
                     0 0 0 1 0"/>
     </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#greyscale)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```


(Immagine sorgente di *Daderot* su Wikimedia Commons)

Filtri sfocatura: Sfocatura messa a fuoco (gaussiana)

```
<svg width="800px" height="600px">
   <defs>
  <filter id="focus-blur" >
    <feDiffuseLighting result = "diffOut" diffuseConstant = "1" lighting-color="white">
       <feSpotLight id="spotlight" x = "500" y = "100" z = "150" pointsAtX = "500" pointsAtY =
```

```
"100" pointsAtZ = "0" specularExponent ="12" limitingConeAngle="70"/>
    </feDiffuseLighting>

     <feColorMatrix in="diffOut" result="alphaMap" type="luminanceToAlpha"/>
     <feComponentTransfer in="alphaMap" result="invertlight">
        <feFuncA type="table" tableValues="1 0 0"/>
     </feComponentTransfer>

     <feGaussianBlur in="invertlight" result="featherspot" stdDeviation="5"/>
     <feComposite operator="xor" result="infocus" in2="SourceGraphic" in="featherspot"/>
     <feGaussianBlur in="SourceGraphic" result="outfocus" stdDeviation="2"/>
     <feComposite operator="over" in="infocus" in2="outfocus"/>
    <feComposite operator="in" in2="SourceGraphic"/>
</filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#focus-blur)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

(Immagine sorgente di *Daderot* su Wikimedia Commons)

Filtri per la manipolazione del colore: Posterizzazione

```
<svg width="800px" height="600px" >
   <defs>
     <filter id="posterize" color-interpolation-filters="sRGB">
     <feComponentTransfer>
       <feFuncR type="discrete" tableValues="0 0.25 0.75 1.0"/>
       <feFuncG type="discrete" tableValues="0 0.25 0.75 1.0"/>
       <feFuncB type="discrete" tableValues="0 0.25 0.75 1.0"/>
     </feComponentTransfer>
     </filter>
   </defs>

   <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/4/42/Andy_Warhol_1975.jpg" x="20px"
y="20px" width="300px" height="600px" preserveAspectRatio="xMinYMin meet" />
   <image filter="url(#posterize)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/4/42/Andy_Warhol_1975.jpg"
x="340px" y="20px" width="300px" height="600px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

Filtri sfocatura: evidenzia sfocatura

Questo filtro seleziona solo le aree ad alta luminanza di un grafico sorgente, offusca il contenuto e compone il contenuto sfocato sopra l'originale.

```
<svg width="800px" height="600px">
    <defs>
        <filter id="highlightblur" color-interpolation-filters="sRGB">
           <feColorMatrix type="luminanceToAlpha" in="SourceGraphic" result="lumMap"/>
           <feComponentTransfer in="lumMap" result="highlightMask">
              <feFuncA type="discrete" tableValues="0 0 0 0 0 0 0 1"/>
           </feComponentTransfer>
           <feComposite operator="in" in="SourceGraphic" in2="highlightMask"
result="highlights"/>
           <feGaussianBlur in="highlights" stdDeviation="3" result="highBlur"/>
           <feComposite operator="over" in="highBlur" in2="SourceGraphic" result="final"/>
        </filter>
     </defs>

    <image filter="url(#highlightblur)" x="0" y="-40" width="780" height="600"
preserveAspectRatio="true"
xlink:href="http://i554.photobucket.com/albums/jj424/allbowerpower/Christmas%202009/ChristmasTablesett

    />
</svg>
```

Leggi filtri online: https://riptutorial.com/it/svg/topic/3262/filtri

Capitolo 10: interruttore

Osservazioni

`<switch>` è un attributo di elaborazione condizionale. Non impedisce agli elementi di essere referenziati da altri elementi. Nel nostro caso, `<switch>` valuta il valore di **systemLanguage** sugli elementi figlio diretti corrispondenti alla lingua dell'utente. Una volta trovato, il bambino viene reso e gli altri bambini saranno esclusi.

Se la **lingua di sistema** non è specificata, verrà visualizzato il figlio, consentendoci di specificare un fallback.

Informazioni sulla raccomandazione relative al W3C

Examples

Visualizzazione alternativa a seconda della lingua dell'utente

```
<svg xmlns="http://www.w3.org/2000/svg">
  <switch>
    <text systemLanguage="en-UK" x="10" y="10">UK English</text>
    <text systemLanguage="fr" x="10" y="10">Français</text>
    <text systemLanguage="ru" x="10" y="10">Русский</text>
    <text x="10" y="20">English</text> <!-- fallback (if none of the languages match) -->
  </switch>
</svg>
```

Leggi interruttore online: https://riptutorial.com/it/svg/topic/4702/interruttore

Capitolo 11: L'elemento SVG

Examples

viewBox

L'attributo viewBox definisce il sistema di coordinate per un elemento `<svg>` . Ciò consente di modificare facilmente la dimensione e la proporzione relativa di un grafico SVG senza dover regolare la posizione e le dimensioni di ogni singolo elemento disegnato.

```
<!-- stretches a small icon to 60px square -->
<svg viewBox="0 0 16 16" height="60px" width="60px">
    <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.02l4.912 3.57-1.904
5.834h.02l4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

Il codice è simile a questo:

Senza il viewBox, assomiglia a questo:

preserveAspectRatio

`preserveAspectRatio` è un attributo che indica se l'immagine deve essere ridimensionata in modo uniforme. Questo attributo funziona solo se l'elemento `<svg>` ha anche un `viewBox` .

Il valore predefinito è `xMidYMid` , che mantiene le proporzioni e centra il percorso all'interno del contenitore SVG:

```
<!-- when not included `preserveAspectRatio` defaults to `xMidYMid` -->
<svg viewBox="0 0 16 16" height="60" width="120">
    <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.02l4.912 3.57-1.904
5.834h.02l4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

Quando `preserveAspectRatio` è impostato su `none` , l'icona si allunga per adattarsi alla casella:

```
<svg viewBox="0 0 16 16" height="60" width="120" preserveAspectRatio="none">
    <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.021 4.912 3.57-1.904
5.834h.021 4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

Esistono molti altri valori per `preserveAspectRatio` , ma questi due sono di gran lunga i più comuni.

preserveAspectRatio: incontra e affetta gli attributi

L'attributo `preserveAspectRatio` ha un parametro facoltativo: `meet` | `slice` . Il comportamento predefinito è `meet` che si estende il contenuto sia nella dimensione xe y fino a riempire la larghezza ***o*** altezza della viewBox. L'alternativa - `slice` di mantenere le proporzioni del contenuto, ma ridimensiona il grafico finché riempie ***sia*** la larghezza e l'altezza del viewBox (clipping il contenuto che trabocca il viewBox).

Questo è l'esempio che usa `slice`

```
<svg viewBox="0 0 16 16" height="60px" width="120px" preserveAspectRatio="xMinYMin slice">
<path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.021 4.912 3.57-1.904
5.834h.021 4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
```

che esegue il rendering come:

e lo stesso esempio usando `meet`

```
    <svg viewBox="0 0 16 16" height="60px" width="120px" preserveAspectRatio="xMinYMin meet">
    <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.021 4.912 3.57-1.904
5.834h.021 4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

che esegue il rendering come:

Leggi L'elemento SVG online: https://riptutorial.com/it/svg/topic/6923/l-elemento-svg

Capitolo 12: Linea

Parametri

Attributo	Descrizione
x1	Posizione orizzontale dell'inizio della linea.
Y1	Posizione verticale dell'inizio della linea.
x2	Posizione orizzontale della fine della linea.
y2	Posizione verticale della fine della linea.
ictus	Colore della linea.
stroke-width	Larghezza della linea.
ictus-opacità	Opacità della linea.
ictus dasharray	Schema di tratteggio per la linea
ictus linecap	Come la linea finisce render

Osservazioni

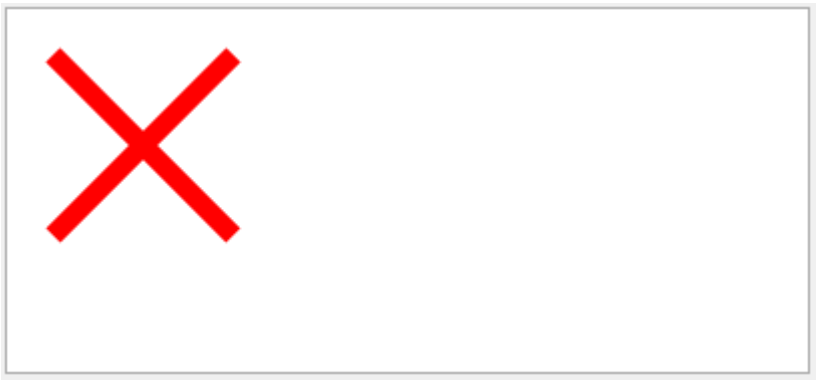
Informazioni dettagliate sull'elemento 'linea' di SVG sono disponibili nella Raccomandazione W3C per SVG .

Examples

Disegna una croce usando linee rosse diagonali

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <line x1="10" y1="10" x2="100" y2="100" stroke="red" stroke-width="10" />
    <line x1="100" y1="10" x2="10" y2="100" stroke="red" stroke-width="10" />
</svg>
```

Risultato:



Disegno tratteggiato con stroke-dasharray

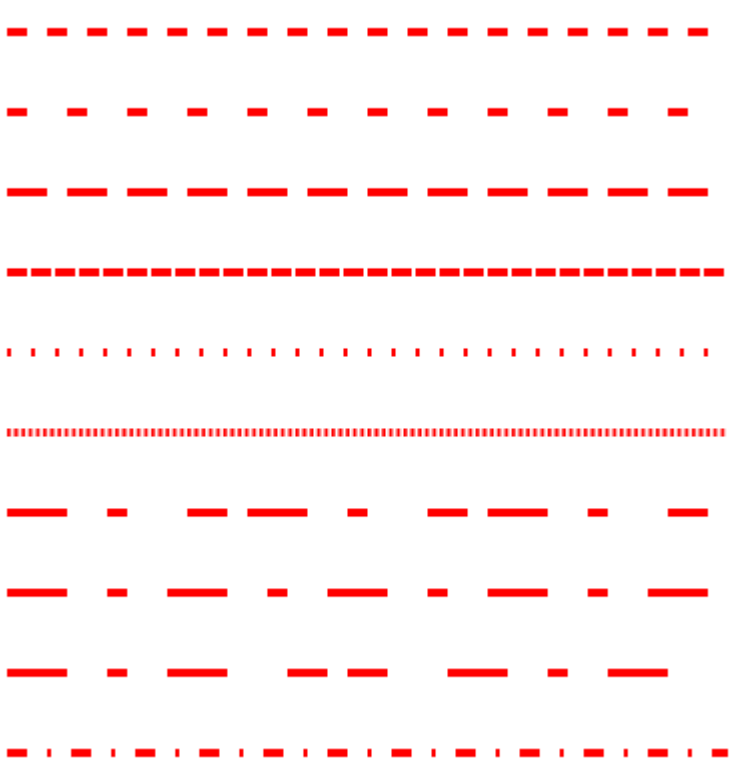
```
<svg width="400px" height="400px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <line x1="10" y1="10" x2="300" y2="10" stroke="red" stroke-width="10" stroke-
dasharray="20,2,5,2"/>
</svg>
```

Risultato

Diversi esempi di stroke-dasharray:

```
<svg width="200" height="200" viewBox="0 0 200 200" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <line stroke-dasharray="5, 5"              x1="10" y1="10" x2="190" y2="10" />
  <line stroke-dasharray="5, 10"             x1="10" y1="30" x2="190" y2="30" />
    <line stroke-dasharray="10, 5"             x1="10" y1="50" x2="190" y2="50" />
    <line stroke-dasharray="5, 1"              x1="10" y1="70" x2="190" y2="70" />
    <line stroke-dasharray="1, 5"              x1="10" y1="90" x2="190" y2="90" />
    <line stroke-dasharray="0.9"               x1="10" y1="110" x2="190" y2="110" />
    <line stroke-dasharray="15, 10, 5"         x1="10" y1="130" x2="190" y2="130" />
    <line stroke-dasharray="15, 10, 5, 10"     x1="10" y1="150" x2="190" y2="150" />
    <line stroke-dasharray="15, 10, 5, 10, 15" x1="10" y1="170" x2="190" y2="170" />
    <line stroke-dasharray="5, 5, 1, 5"        x1="10" y1="190" x2="190" y2="190" />
    <style><![CDATA[
    line{
        stroke: red;
        stroke-width: 2;
    }
    ]]></style>
 </svg>
```

Risultato:



Alternative di line cap utilizzando stroke-line

```
<svg width="600px" height="400px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">

    <line x1="10" y1="20" x2="300" y2="20" stroke="red" stroke-width="20" stroke-
linecap="butt"/>
      <text x="320" y="20">stroke-linecap="butt" (default)</text>
    <line x1="10" y1="70" x2="300" y2="70" stroke="red" stroke-width="20" stroke-
linecap="round"/>
      <text x="320" y="70">stroke-linecap="round"</text>
    <line x1="10" y1="120" x2="300" y2="120" stroke="red" stroke-width="20" stroke-
linecap="square"/>
      <text x="320" y="120">stroke-linecap="square"</text>
</svg>
```

Risultato

Leggi Linea online: https://riptutorial.com/it/svg/topic/3034/linea

Capitolo 13: marcatore

Sintassi

- `<marker` viewBox = " *xy width height* " refX = " *xoffset* " refY = " *yoffset* " orient = " *orientation* " ... *parametri opzionali* >
- *... elementi che disegnano il marker ...*
- `</marker` >
- < *elementname* marker-start = "url (# *markerid*)" /> applica un marker all'inizio di un elemento
- < *elementname* marker-mid = "url (# *markerid*)" /> applica un marcatore al centro di un segmento di un elemento
- < *elementname* marker-end = "url (# *markerid*)" /> applica un marcatore alla fine di un elemento
- I marker possono essere applicati agli elementi `<line>` , `<polyline>` , `<polygon>` e `<path>`

Parametri

Parametro	Dettagli
viewBox	Specifica il sistema di unità per gli elementi che disegnano il marcatore
reFX	Distanza dall'asse x del sistema di coordinate per disegnare il marcatore deve essere spostato rispetto al punto di disegno predefinito. Il valore predefinito è 0.
refY	Distanza dall'asse y del sistema di coordinate per disegnare il marcatore deve essere spostato rispetto al punto di disegno predefinito. Il valore predefinito è 0.
Oriente	I valori sono `auto` o `angle in degrees` e specifica la rotazione applicata al marker. Viene applicato dopo che sono state eseguite tutte le altre regolazioni delle coordinate (viewBox, preserveAspectRaio e refX, refY). Il valore predefinito è 0. Il calcolo dell'angolo per `auto` è complesso: vedere le specifiche SVG per i dettagli.
markerUnits	`strokeWidth` o `userSpaceOnUse` . Impostazioni predefinite per `strokeWidth` .
markerWidth	Larghezza dell'indicatore in markerUnits. Il valore predefinito è 3.
markerHeight	Altezza del marker in markerUnits. Il valore predefinito è 3

Osservazioni

Scripting: gli elementi dei marker renderizzati non sono esposti nel DOM, quindi è impossibile regolare le proprietà o gli elementi per specifici indicatori di rendering (anche se è completamente possibile scrivere l'elemento marker definito).

La proprietà di `overflow` dell'elemento marker viene automaticamente impostata su `hidden` . Questo è ciò che taglia qualsiasi disegno che trabocca dalla tessera marcatore. Questo può essere esplicitamente impostato su `visible` in CSS. A partire da luglio 2016, Chrome non supporta i marcatori con `overflow: visible` - ma una soluzione alternativa è impostare un filtro sull'elemento marker, che sembra disabilitare il clipping overflow.

I filtri possono essere applicati agli elementi all'interno di un marker. Anche se non sono esplicitamente permessi nelle specifiche, i filtri sembrano funzionare anche se specificati sull'elemento marker stesso.

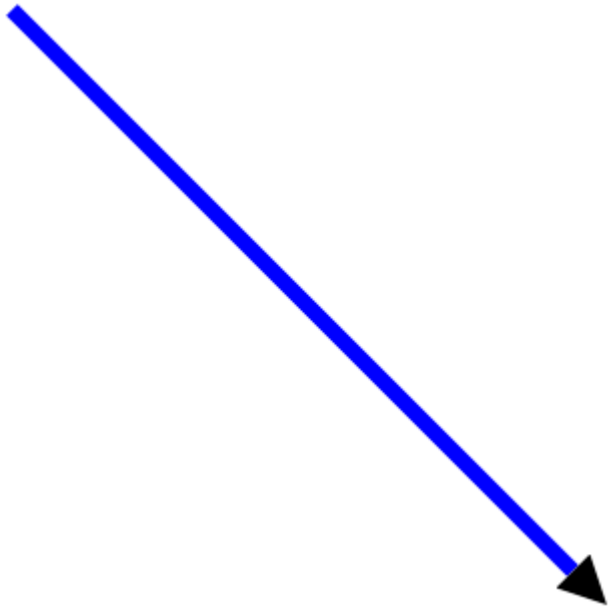
Per maggiori dettagli sull'elemento marcatore, vedere la sezione marcatori nelle specifiche SVG 1.1 .

Examples

Marcatore di base

Questo è un esempio di un marcatore finale specificato con un numero minimo di parametri. Si noti che il colore del tratto dell'elemento originale non è ereditato dal marcatore.

```
<svg width="800px" height="600px">
<defs>
      <marker id="examplemarker"
        viewBox="0 0 10 10"
        refX="0" refY="5"
        orient="auto">
        <path d="M 0 0 L 10 5 L 0 10 z" />
      </marker>
  </defs>

  <line x1="20" y1="20" x2="300" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#examplemarker)" />
</svg>
```

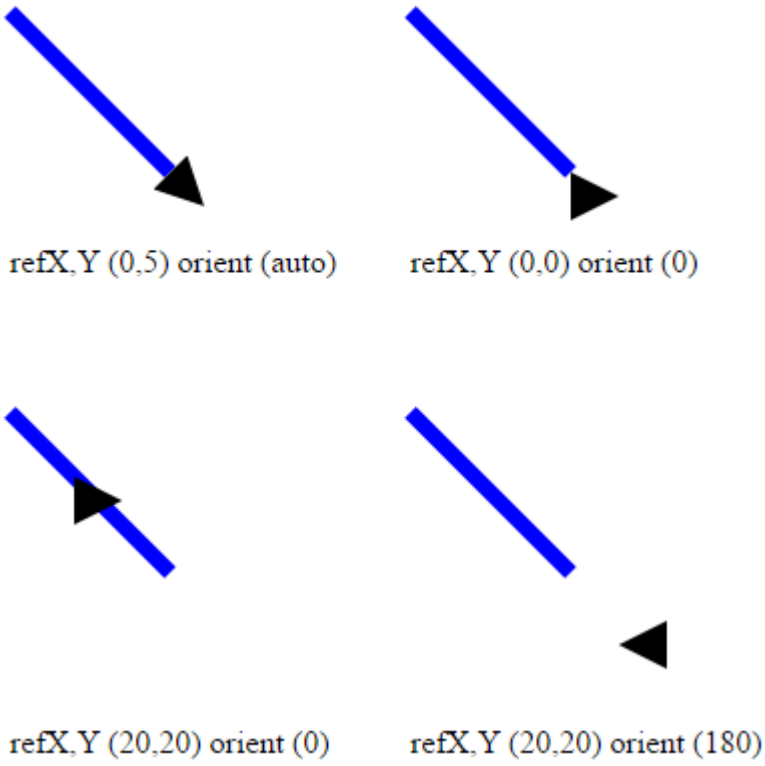


Effetti di valori alternativi per refX, refY e orient

Gli offset dell'asse per disegnare il marker che sono specificati da `refX` e `refY` vengono applicati *prima* della rotazione specificata dal parametro `orient` . Qui sotto puoi vedere gli effetti di varie combinazioni di `orient` e `refX` , fare `refY` a questo.

```
<svg width="800px" height="600px">
<defs>
      <marker id="marker1"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" >
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker2"
      viewBox="0 0 10 10" refX="0" refY="0" orient="0" >
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker3"
      viewBox="0 0 10 10" refX="20" refY="20" orient="0" >
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker4"
      viewBox="0 0 10 10" refX="20" refY="20" orient="180" >
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
  </defs>

  <line  x1="20" y1="20" x2="100" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker1)" />

  <text x="20" y="150"> refX,Y  (0,5) orient (auto) </text>

  <line  x1="220" y1="20" x2="300" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker2)" />

    <text x="220" y="150"> refX,Y  (0,0) orient (0) </text>

  <line  x1="20" y1="220" x2="100" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker3)" />
```

```
      <text x="20" y="390"> refX,Y  (20,20) orient (0) </text>

  <line  x1="220" y1="220" x2="300" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker4)" />

      <text x="220" y="390"> refX,Y  (20,20) orient (180) </text>
</svg>
```

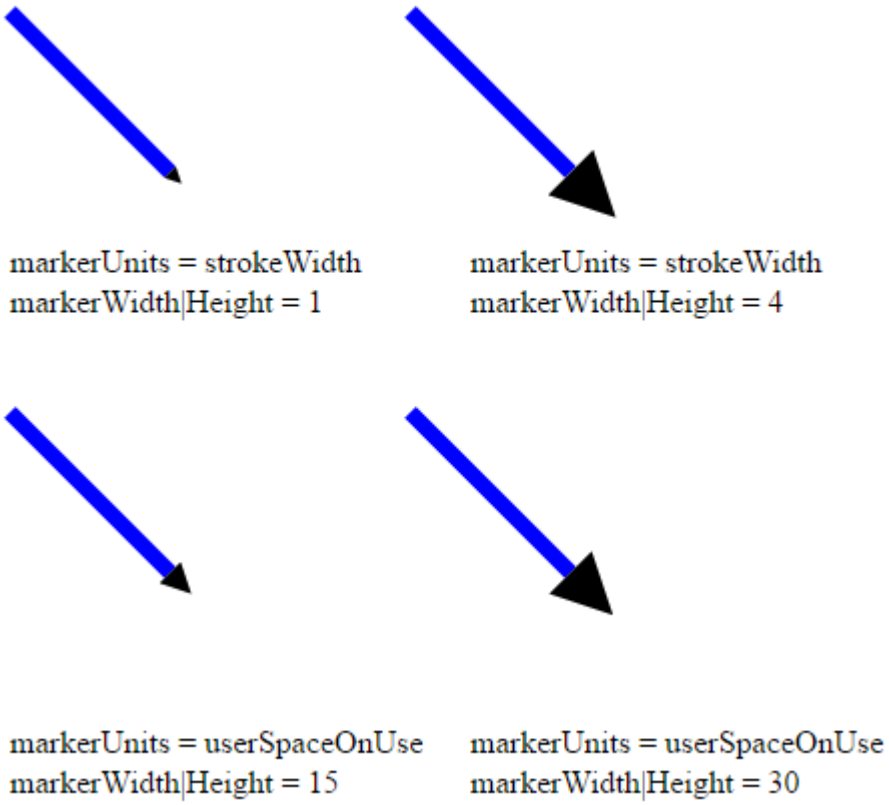


Effetti di valori alternativi per markerUnits, markerWidth, markerHeight

L'impostazione predefinita per gli indicatori di disegno consiste nell'utilizzare la larghezza del tratto dell'elemento chiamante, ma è possibile specificare esplicitamente che i marcatori vengano disegnati utilizzando il sistema di unità per l'elemento al quale il marcatore viene applicato specificando `markerUnits="userSpaceOnUse"` . I marker sono disegnati in una casella 3x3 markerUnits (3 strokewidths se markerUnits non è specificato). Ma la larghezza e l'altezza della scatola possono essere specificate esplicitamente con `markerHeight` e `markerWidth` . Vedi sotto per gli effetti di varie combinazioni di `markerUnits` , `markerHeight` e `markerWidth` .

```
<svg width="800px" height="600px">
<defs>
      <marker id="marker1"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="strokeWidth"
markerWidth="1" markerHeight="1">
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker2"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="strokeWidth"
markerWidth="4" markerHeight="4">
      <path d="M 0 0 L 10 5 L 0 10 z" />
```

```
                                    </marker>
      <marker id="marker3"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="userSpaceOnUse"
markerWidth="15" markerHeight="15">
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
      <marker id="marker4"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="userSpaceOnUse"
markerWidth="30" markerHeight="30">
      <path d="M 0 0 L 10 5 L 0 10 z" />
                                    </marker>
  </defs>

  <line  x1="20" y1="20" x2="100" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker1)" />
    <text x="20" y="150"> markerUnits = strokeWidth </text>
    <text x="20" y="170"> markerWidth|Height = 1 </text>

  <line  x1="220" y1="20" x2="300" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker2)" />
    <text x="250" y="150"> markerUnits = strokeWidth </text>
    <text x="250" y="170"> markerWidth|Height = 4 </text>

  <line  x1="20" y1="220" x2="100" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker3)" />
      <text x="20" y="390"> markerUnits = userSpaceOnUse </text>
      <text x="20" y="410"> markerWidth|Height = 15 </text>

  <line  x1="220" y1="220" x2="300" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker4)" />
      <text x="250" y="390"> markerUnits = userSpaceOnUse </text>
      <text x="250" y="410"> markerWidth|Height = 30 </text>
</svg>
```

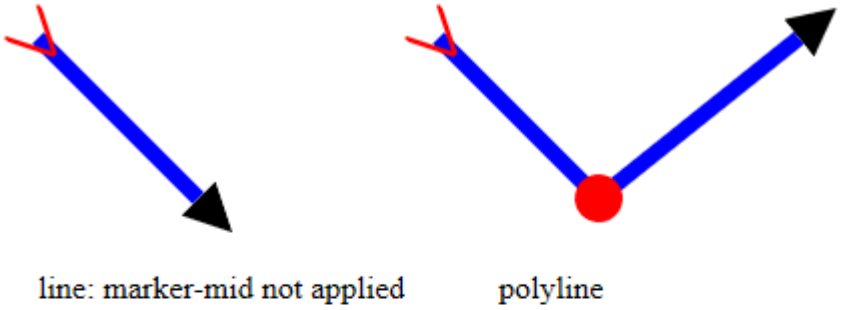


Marcatori di inizio, metà e fine in linea, polilinea, poligono e percorso

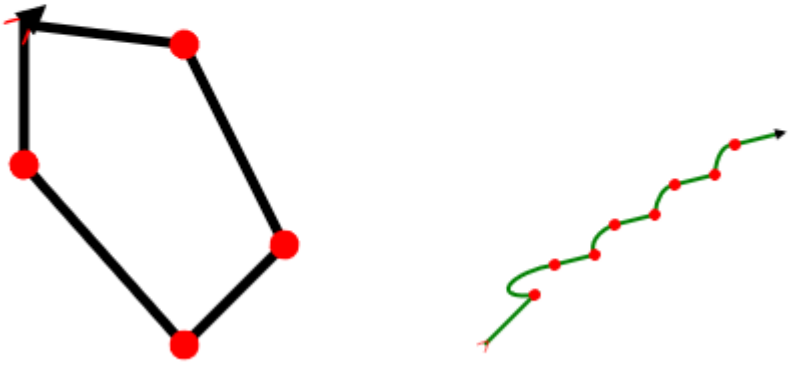
Gli elementi possono specificare separatamente i marcatori di inizio, metà e fine. Di seguito sono riportati alcuni esempi di indicatori di inizio, metà e fine per tutti gli elementi che possono essere contrassegnati. Tieni presente che Chrome al momento (luglio 2016) non calcola correttamente l'orientamento automatico per i marker di inizio e fine per i poligoni (bug # 633012) e inoltre non colloca correttamente i marker medi per i segmenti del tracciato dell'arco (bug # 583097)

```
<svg width="800px" height="600px">
<defs>
      <marker id="red-chevron"
      viewBox="0 0 10 10" refX="5" refY="5" orient="auto" >
         <path d="M 0 0 L 10 5 L 0 10" fill="none" stroke="red" />
      </marker>

      <marker id="black-arrow"
      viewBox="0 0 10 10" refX="0" refY="5" orient="auto">
         <path d="M 0 0 L 10 5 L 0 10 z" />
      </marker>

      <marker id="red-circle"
      viewBox="0 0 10 10" refX="5" refY="5" orient="auto" >
         <circle fill="red" cx="5" cy="5" r="5" />
      </marker>
  </defs>

  <line  x1="20" y1="20" x2="100" y2="100" stroke-width="8" stroke="blue" marker-
start="url(#red-chevron)" marker-end="url(#black-arrow)" marker-mid="url(#red-circle)" />
  <text x="20" y="150"> line: marker-mid not applied</text>
```

```
  <polyline  points="220,20 300,100 400,20" fill="none" stroke-width="8" stroke="blue" marker-
start="url(#red-chevron)" marker-end="url(#black-arrow)" marker-mid="url(#red-circle)" />
    <text x="250" y="150"> polyline </text>

  <polygon  points="20,190 100,200 150,300 100,350 20,260" marker-start="url(#red-chevron)"
marker-end="url(#black-arrow)" marker-mid="url(#red-circle)" fill="none" stroke-width="5"
stroke="black" />
   <text x="20" y="390"> polygon: end/start overlap </text>

  <path d="M250,350 l 25,-25
        a15,5 -16 0,1 10,-15 l 20,-5
        a15,10 -16 0,1 10,-15 l 20,-5
        a15,25 -16 0,1 10,-15 l 20,-5
        a15,35 -16 0,1 10,-15 l 20,-5"
        fill="none" stroke="green" stroke-width="2" marker-start="url(#red-chevron)" marker-
end="url(#black-arrow)" marker-mid="url(#red-circle)"  />
      <text x="250" y="390"> path with arc segments </text>
</svg>
```



line: marker-mid not applied polyline



polygon: end/start overlap path with arc segments

Leggi marcatore online: https://riptutorial.com/it/svg/topic/4839/marcatore

Capitolo 14: maschera

introduzione

L'elemento `mask` ti consente di "ritagliare" con bordi sfumati. Puoi comporre maschere da elementi multipli, incluso il testo. Tutto di una maschera che è bianca sarà completamente opaca. Tutto ciò che è nero sarà completamente trasparente. I valori tra bianco e nero saranno semitrasparenti.

Osservazioni

Essere consapevoli del fatto che le maschere sono un'operazione costosa computazionale. Il calcolo deve essere fatto per ogni pixel nell'area della maschera. Quindi mantieni l'area della maschera più piccola possibile.

Examples

maschera di base

Un rect verde con un buco rotondo nel mezzo che mostra lo sfondo dell'immagine sotto.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <mask id="myMask">
        <rect x="0" y="0" width="100" height="100" fill="white"/>
        <circle cx="50" cy="50" r="45" fill="black"/>
    </mask>
    <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
    <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

esempio complesso con testo e forme

Un rettangolo verde con una maschera complessa applicata ad esso che mostra l'immagine di sfondo.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <mask id="myMask0">
        <circle cx="50" cy="50" r="30" fill="white"/>
    </mask>
    <mask id="myMask">
        <rect x="0" y="0" width="100" height="100" fill="white"/>
        <text x="5" y="60" font-size="40">Mask</text>
        <circle cx="50" cy="50" r="30" fill="black"/>
        <text x="5" y="60" font-size="40" mask="url(#myMask0)" fill="white">Mask</text>
    </mask>
    <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
```

```
    <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

semi trasparenza

un rettangolo verde (di nuovo ...) con 4 fori creati utilizzando 4 valori in scala di grigi con 4 diverse opacità.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <mask id="myMask">
        <rect x="0" y="0" width="100" height="100" fill="white"/>
        <circle cx="25" cy="25" r="20" fill="black"/>
        <circle cx="75" cy="25" r="20" fill="#333"/>
        <circle cx="25" cy="75" r="20" fill="#666"/>
        <circle cx="75" cy="75" r="20" fill="#999"/>
    </mask>
    <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
    <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

una maschera con un gradiente

Un rect verde con un buco nel mezzo, con bordi morbidi.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <radialGradient id="rg">
      <stop offset="0" stop-color="black"/>
      <stop offset="1" stop-color="white"/>
    </radialGradient>
    <mask id="myMask">
        <rect x="0" y="0" width="100" height="100" fill="white"/>
        <circle cx="50" cy="50" r="45" fill="url(#rg)"/>
    </mask>
    <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
    <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

Leggi maschera online: https://riptutorial.com/it/svg/topic/8143/maschera

Capitolo 15: Patterns

Parametri

parametro	descrizione
patternUnits	il sistema di coordinate degli attributi del modello è objectBoundingBox (predefinito) o userSpaceOnUse
patternContentUnits	il sistema di coordinate dei contenuti del pattern sia objectBoundingBox o userSpaceOnUse (predefinito)
patternTransform	la trasformazione da applicare ai contenuti del pattern
X	l'offset x del modello (l'impostazione predefinita è zero)
y	l'offset y del modello (il valore predefinito è zero)
larghezza	la larghezza del modello (richiesto)
altezza	l'altezza del modello (richiesto)
xlink: href	link a un altro pattern che fornisce alcuni attributi o contenuti
preserveAspectRatio	se le proporzioni del modello devono essere preservate

Osservazioni

Per impostazione predefinita, il motivo verrà affiancato impostando la parte centrale dell'unità motivo nell'angolo in alto a sinistra della forma.

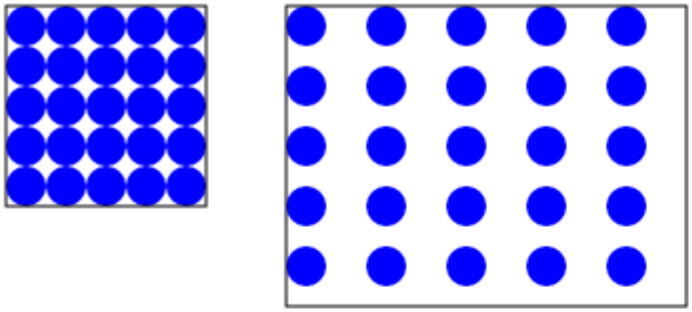
Examples

Esempio di modello con unità objectBoundingBox

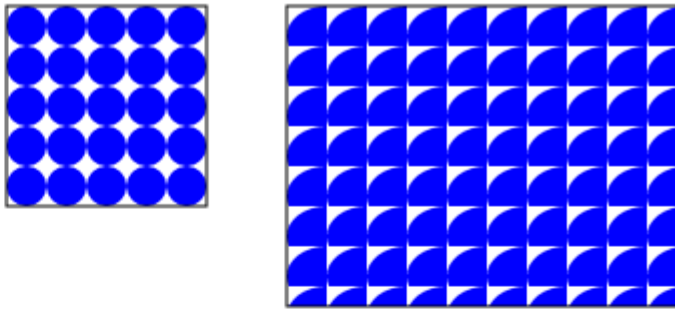
```
<svg width="400" height="400">
<defs>
  <pattern id="pattern1" width="0.2" height="0.2" patternUnits="objectBoundingBox">
      <circle cx="10" cy="10" r="10" fill="#0000ff" />
  </pattern>
</defs>

<rect x="10" y="10" width="100" height="100" stroke="black" fill="url(#pattern1)"/>
</svg>
```

Copertura del pattern con combinazioni di patternUnits e patternContentUnits

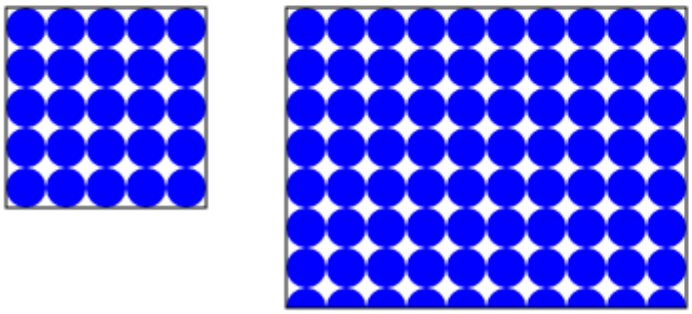
I pattern SVG si comportano in modo significativamente diverso rispetto alle immagini di sfondo CSS quando si riempiono forme equivalenti. Questo può portare a sorprese significative per i neofiti di SVG. Di seguito sono riportati alcuni esempi di pattern definiti in tutte le possibili combinazioni di patternUnits e patternContentUnits, che mostrano come queste impostazioni influenzano il comportamento di riempimento.



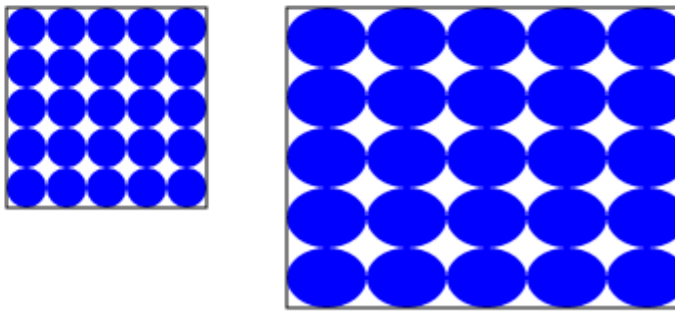
patternUnits="objectBoundingBox" (20% of shape)
patternContentUnits="userSpaceOnUse" (20px circle)
(Units used by default)



patternUnits="userSpaceOnUse" (10px square box)
patternContentUnits="objectBoundingBox"(radius=



patternUnits="userSpaceOnUse" (10px square box)
patternContentUnits="userSpaceOnUse" (20px circle)



patternUnits="objectBoundingBox" (20% of shape)
patternContentUnits="objectBoundingBox"(radius=

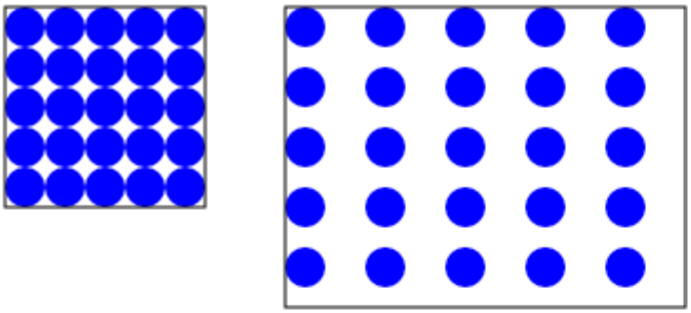
```
<svg width="800px" height="800px">
<defs>
<pattern id="pattern1" x="0" y="0" width="0.2" height="0.2"  patternUnits="objectBoundingBox"
patternContentUnits="userSpaceOnUse">
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

    <pattern id="pattern2" x="10" y="10" width="20" height="20"  patternUnits="userSpaceOnUse"
patternContentUnits="objectBoundingBox">
      <circle cx=".1" cy=".1" r="0.1" fill="blue" />
  </pattern>

    <pattern id="pattern3" x="10" y="10" width="20" height="20"  patternUnits="userSpaceOnUse"
patternContentUnits="userSpaceOnUse">
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

    <pattern id="pattern4" x="0" y="0" width="0.2" height="0.2"
```

```
patternUnits="objectBoundingBox" patternContentUnits="objectBoundingBox">
      <circle cx=".1" cy=".1" r="0.1" fill="blue" />
  </pattern>
</defs>

<rect x="10" y="10" width="100" height="100" stroke="black" fill="url(#pattern1)"/>
<rect x="150" y="10" width="200" height="150" stroke="black" fill="url(#pattern1)"/>
  <text x="10" y="200">patternUnits="objectBoundingBox" (20% of shape)</text>
  <text x="10" y="220">patternContentUnits="userSpaceOnUse" (20px circle) </text>
  <text x="10" y="240" stroke="blue" stroke-width="1">(Units used by default)</text>

<rect x="10" y="310" width="100" height="100" stroke="black" fill="url(#pattern3)"/>
<rect x="150" y="310" width="200" height="150" stroke="black" fill="url(#pattern3)"/>
  <text x="10" y="500">patternUnits="userSpaceOnUse" (10px square box)</text>
  <text x="10" y="520">patternContentUnits="userSpaceOnUse" (20px circle) </text>

<rect x="410" y="10" width="100" height="100" stroke="black" fill="url(#pattern2)"/>
<rect x="550" y="10" width="200" height="150" stroke="black" fill="url(#pattern2)"/>
  <text x="410" y="200">patternUnits="userSpaceOnUse" (10px square box)</text>
  <text x="410" y="220">patternContentUnits="objectBoundingBox"(radius="10%") </text>

<rect x="410" y="310" width="100" height="100" stroke="black" fill="url(#pattern4)"/>
<rect x="550" y="310" width="200" height="150" stroke="black" fill="url(#pattern4)"/>
  <text x="410" y="500">patternUnits="objectBoundingBox" (20% of shape)</text>
  <text x="410" y="520">patternContentUnits="objectBoundingBox"(radius="10%")  </text>

</svg>
```

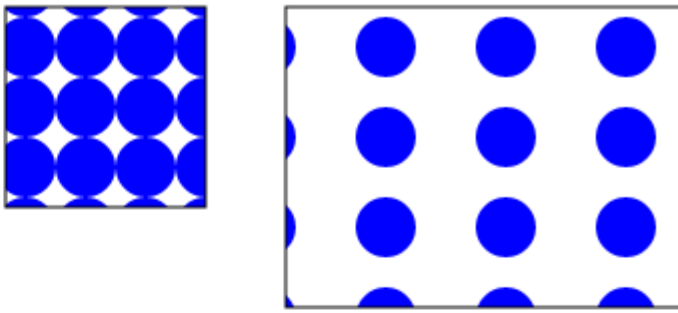
patternTransform esempi

```
<svg width="800px" height="800px">
<defs>
<pattern id="pattern1" x="0" y="0" width="0.2" height="0.2" >
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

  <pattern id="pattern2" x="0" y="0" width="0.2" height="0.2" patternTransform="scale(1.5)">
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

  <pattern id="pattern3" x="0" y="0" width="0.2" height="0.2"   patternTransform="skewX(45)">
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>

  <pattern id="pattern4" x="0" y="0" width="0.2" height="0.2"  patternTransform="matrix(1.5,-
.70,.10,1.1,-30,10)">
      <circle cx="10" cy="10" r="10" fill="blue" />
  </pattern>


</defs>

<rect x="10" y="10" width="100" height="100" stroke="black" fill="url(#pattern1)"/>
<rect x="150" y="10" width="200" height="150" stroke="black" fill="url(#pattern1)"/>
  <text x="10" y="200">Original</text>

<rect x="410" y="10" width="100" height="100" stroke="black" fill="url(#pattern2)"/>
<rect x="550" y="10" width="200" height="150" stroke="black" fill="url(#pattern2)"/>
  <text x="410" y="200">patternTransform="scale(1.5)"</text>
```

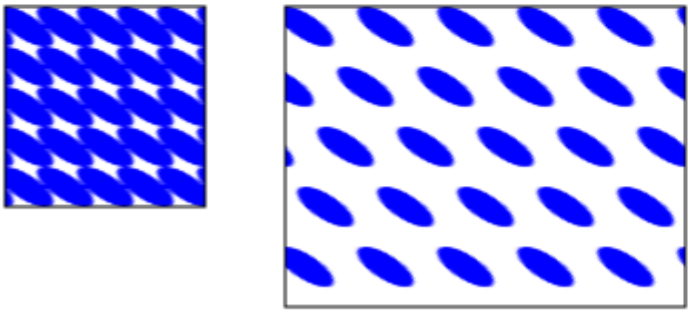
```
<rect x="10" y="310" width="100" height="100" stroke="black" fill="url(#pattern3)"/>
<rect x="150" y="310" width="200" height="150" stroke="black" fill="url(#pattern3)"/>
  <text x="10" y="500">patternTransform="skewX(45)"</text>

<rect x="410" y="310" width="100" height="100" stroke="black" fill="url(#pattern4)"/>
<rect x="550" y="310" width="200" height="150" stroke="black" fill="url(#pattern4)"/>
  <text x="410" y="500">patternUnits="matrix(1.5,-.70,.10,1.1,-30,10)"</text>

</svg>
```



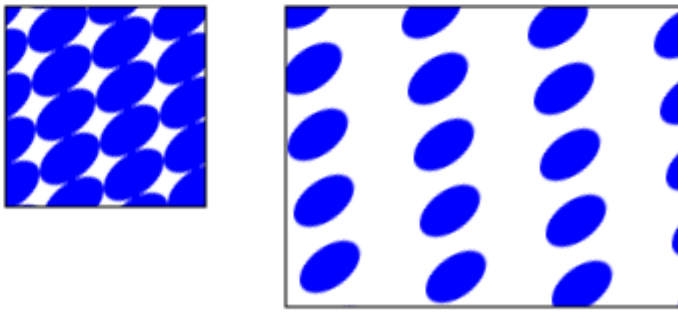
Original



patternTransform="scale(1.5)"



patternTransform="skewX(45)"



patternUnits="matrix(1.5,-.70,.10,1.1,-30,10)"

Leggi Patterns online: https://riptutorial.com/it/svg/topic/3251/patterns

Capitolo 16: percorsi

introduzione

I percorsi sono l'elemento più flessibile di SVG. Un percorso è una serie di curve di Bézier cubiche o quadratiche, disposte in spline collegate. Un percorso può essere aperto o chiuso in un ciclo o può essere complesso con diversi sottocomponenti. Se un percorso non è semplice, la regola di riempimento è importante per decidere quali aree sono all'interno o all'esterno del percorso.

I percorsi verranno normalmente generati dagli editor automatici. I percorsi tipicamente quadratici vengono utilizzati per i caratteri e i percorsi cubici per le illustrazioni.

Parametri

Attributi / parametri	Descrizione
d	Definisce una sequenza di comandi di disegno che creano la forma. ad es. d = "M 50,60 L50,60". I comandi di disegno maiuscoli indicano le coordinate assolute. I comandi di disegno in lettere minuscole indicano le coordinate relative.
(...)	**Comandi di disegno**
m / M	Sposta la posizione del disegno corrente su XY d = "M XY"
l / L	Disegna una linea su X, Y d = "L XY"
v / V	Disegna una linea verticale su Y d = "V Y"
h / H	Disegna una linea orizzontale a X d = "H X"
aa	Disegna un arco su X, Y con un raggio implicito di Rx e Ry e una rotazione specificata dalla rotazione dell'asse X. I grandi flag arco e sweep selezionano quale dei 4 possibili archi che soddisfano questi vincoli dovrebbero essere disegnati. d = "A Rx Ry X-axis-rotation (degrees) flag dell'arco di grandi dimensioni (0/1) sweep-flag (0/1) X, Y".
q / Q	Disegna la curva quadratica di bezier su X, Y utilizzando il punto di controllo X1Y1 d = "X1Y1 X Y"
t / T	Disegna una curva di bezier quadratica stenografica (il punto di controllo viene calcolato come riflesso del punto di controllo del precedente comando di disegno q / Q attraverso la posizione di disegno corrente)
c / C	Disegna una curva cubica di bezier su X, Y usando i punti di controllo X1, Y1

Attributi / parametri	Descrizione
	e X2, Y2 d = "C X1Y1, X2Y2, XY"
s / S	Disegna una curva Bezier a forma di cubica (il primo punto di controllo viene calcolato come riflesso del secondo punto di controllo del precedente comando di disegno c / C attraverso la posizione di disegno corrente).
- z / Z	Chiudi il tracciato disegnando una linea all'inizio del percorso (o dei percorsi se un'altra z è stata utilizzata in precedenza)
(...)	*(fine della lista)*
lunghezza del tragitto	(Facoltativo) Consente all'autore di specificare un valore nominale del percorso che verrà utilizzato per la calibrazione in altri calcoli, ad esempio per il testo lungo un percorso
Parametri di corsa	*comune tra tutti gli elementi di forma e disegno*
ictus	Colore del percorso
stroke-width	Larghezza del percorso

Osservazioni

Informazioni dettagliate sull'elemento del `path` SVG possono essere trovate nella Raccomandazione W3C per SVG .

Examples

Disegna una linea blu diagonale usando il comando L path

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <path d="M 10,10 L 100,50" stroke="blue" stroke-width="5" />
</svg>
```

Risultato:

Disegna una linea arancione orizzontale usando il comando di disegno H.

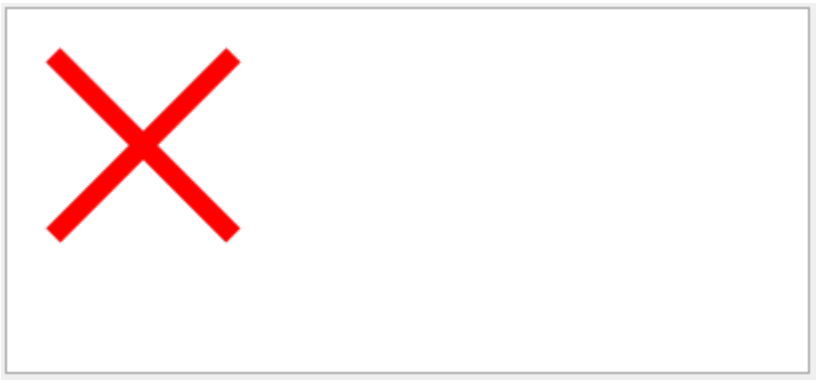
```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <path d="M 10,10 H 200" stroke="orange" stroke-width="5" />
</svg>
```

Risultato:

Disegna una croce rossa usando i comandi di percorso l (linea relativa)

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <path d="M 10,10 l 90,90 M 100,10 l -90,90" stroke="red" stroke-width="10" />
</svg>
```

Risultato:



Disegna una linea verde verticale usando il comando percorso V

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <path d="M 10,10 V 200" stroke="green" stroke-width="5" />
</svg>
```

Risultato:

Leggi percorsi online: https://riptutorial.com/it/svg/topic/2397/percorsi

Capitolo 17: pointer-events

introduzione

Con la proprietà `pointer-events` , puoi controllare quale parte del tuo disegno reagirà agli eventi del puntatore.

Examples

nessuna

il caso d'uso più comune è impostare gli eventi puntatore su `none` per impedire a determinate forme o a tutto il disegno di catturare gli eventi del mouse e lasciare che le forme sotto di loro ricevano gli eventi.

Se passi con il mouse sopra l'area in cui il cerchio rosso si sovrappone al cerchio blu, il cerchio blu continuerà a ricevere gli eventi del mouse, poiché gli eventi puntatore non sono impostati su `none`

```
<svg viewBox="0 0 150 100">
    <style>
        .target:hover{fill:green}
    </style>
    <circle class="target" cx="50" cy="50" r="50" fill="blue"/>
    <circle cx="100" cy="50" r="50" fill="red" pointer-events="none"/>
</svg>
```

riempire

Setting `pointer-events="fill"` ti permette di ricevere eventi del mouse su una forma anche se il suo riempimento è impostato su `none`

```
<svg viewBox="0 0 100 100">
    <style>
        circle:hover{fill:green}
    </style>
    <circle class="target" cx="50" cy="50" r="50" fill="none"/>
</svg>
```

Leggi pointer-events online: https://riptutorial.com/it/svg/topic/8166/pointer-events

Capitolo 18: polilinea

Sintassi

- `<polyline points="10,5 25,15 20,10" />`

Parametri

Parametro	Dettagli
punti	L'attributo punti definisce un elenco di punti. Ogni punto è definito dall'asse e da una coordinata nel sistema di coordinate dell'utente.
stroke-width	Larghezza del tratto
ictus-opacità	Opacità di ictus
ictus dasharray	(Opzionale) Specifica il motivo del trattino per il tratto
ictus linecap	(Facoltativo) Specifica se la fine della linea deve essere a filo, rotonda o quadrata ("testa" (predefinito) / "rotondo" / "quadrato")
ictus linejoin	(Opzionale) Specifica il modo in cui i segmenti di linea devono essere uniti: smussati, arrotondati o smussati ("mitra" (predefinito) / "rotondo" / "smusso")
ictus miterLimit	(Opzionale) Specifica la dimensione massima di una mitra. I join con giunture che superano questo limite vengono convertiti in un join smussato. Default = "4"

Examples

SVG che include una polilinea

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <polyline points="10,5 25,15 20,10" />
</svg>
```

Polilinee con linee alternative, lancette e mitigini

```
<svg width="600px" height="600px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">

    <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" />
```

```
      <text x="320" y="20">Default drawing stroke</text>

      <g transform="translate(0,150)">
      <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" stroke-linecap="butt" stroke-linejoin="miter" stroke-
miterlimit="2"/>

      <text x="320" y="20">stroke-linecap="butt" (default)</text>
      <text x="320" y="40">stroke-linejoin="miter"(default)</text>
      <text x="320" y="60">stroke-miterlimit="2"</text>
  </g>

  <g transform="translate(0,300)">
      <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" stroke-linecap="round" stroke-linejoin="round" />

      <text x="320" y="20">stroke-linecap="round" </text>
      <text x="320" y="40">stroke-linejoin="round" </text>

  </g>
  <g transform="translate(0,450)">
      <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" stroke-linecap="square" stroke-linejoin="bevel"/>

      <text x="320" y="20">stroke-linecap="square"</text>
      <text x="320" y="40">stroke-linejoin="bevel"</text>
  </g>

</svg>
```

Risultato

Default drawing stroke

stroke-linecap="butt" (default)
stroke-linejoin="miter"(default)
stroke-miterlimit="2"

stroke-linecap="round"
stroke-linejoin="round"

stroke-linecap="square"
stroke-linejoin="bevel"

Leggi polilinea online: https://riptutorial.com/it/svg/topic/3842/polilinea

Capitolo 19: Rettangolo

Parametri

Attributo	Descrizione
X	Posizione orizzontale del rettangolo dal margine sinistro.
y	Posizione verticale del rettangolo dal margine superiore.
larghezza	Larghezza del rettangolo
altezza	Altezza del rettangolo
rx	Raggio orizzontale dell'ellisse utilizzato per arrotondare gli angoli del rettangolo
ry	Raggio verticale dell'ellisse usato per arrotondare gli angoli del rettangolo
ictus	Colore del bordo del rettangolo.
stroke-width	Larghezza del bordo del rettangolo.
riempire	Colore *all'interno del* bordo del rettangolo.

Osservazioni

Informazioni dettagliate sull'elemento 'rect' di SVG sono disponibili nella Raccomandazione W3C per SVG .

Examples

Disegna un rettangolo nero senza riempimento

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="10" y="10" width="50" height="100" stroke="black" stroke-width="5" fill="none" />
</svg>
```

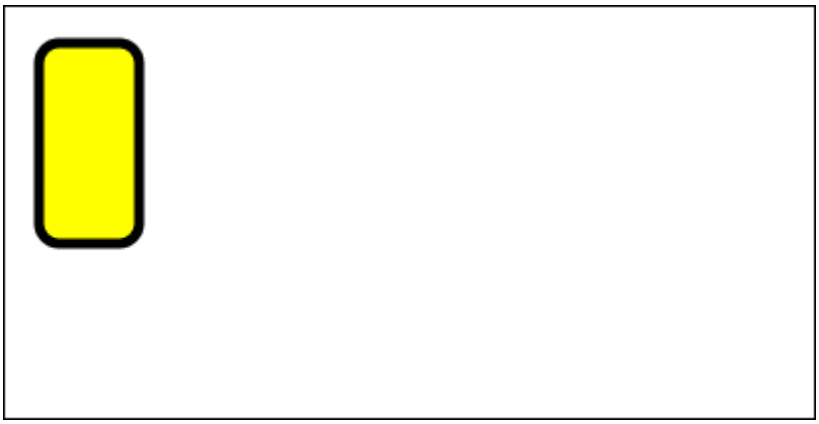
Risultato:

Disegna un rettangolo nero con riempimento giallo e angoli arrotondati

- Gli attributi `width` e `height` indicano le dimensioni del rettangolo. Questi valori sono in pixel per impostazione predefinita
- Il valore di `fill` imposta il colore per il rettangolo. Se non viene specificato alcun valore per il `fill` , per impostazione predefinita viene utilizzato il nero

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="10" y="10" width="50" height="100" rx="10" ry="10" stroke="black" stroke-
width="5" fill="yellow" />
</svg>
```

Risultato:



Leggi Rettangolo online: https://riptutorial.com/it/svg/topic/2993/rettangolo

Capitolo 20: Scripting

Osservazioni

Scripting SVG utilizzando le interfacce DOM native è attualmente (2016) in uno stato di lieve cambiamento. L'attuale standard SVG (1.1) è ben implementato dalla maggior parte dei browser Web principali. Tuttavia, poiché lo standard SVG 2.0 è in fase di sviluppo, alcuni browser hanno iniziato a rimuovere le funzionalità SVG 1.1 che saranno obsolete in 2.0. Puoi vedere un elenco completo delle modifiche proposte da SVG 1.1 a SVG 2.0 nell'Appendice L di SVG 2.0 .

Sostituzione di `pathSegList` e altro utilizzo di `SVGPathSeg`

In SVG 1.1 gli elementi `<path>` sono definiti per avere una proprietà `pathSegList` che dà accesso a una rappresentazione nativa di tutti i comandi di percorso . Google Chrome v48 ha rimosso questa proprietà alla fine del 2015, in preparazione di una proposta di sostituzione in SVG 2.0 . Fino a quando non viene aggiunto il supporto SVG 2.0, è necessario utilizzare un polyfill per ripristinare la funzionalità 1.1 o per implementare l'API 2.0 proposta .

Sostituire `getTransformToElement()`

Anche Chrome v48 ha rimosso il metodo `SVGGraphicsElement.getTransformToElement()` . Esiste un semplice polyfill per implementare il vecchio metodo .

Examples

Creare un elemento

Il modo più semplice per comprendere la creazione e la modifica degli elementi SVG è di operare sugli elementi usando le interfacce DOM Level 2 Core , come faresti con HTML o XML.

È imperativo che gli *elementi* creati da JavaScript siano creati nello stesso spazio dei nomi dichiarato nell'elemento SVG - in questo esempio: " http://www.w3.org/2000/svg ". Tuttavia, quasi tutti gli *attributi* degli elementi SVG non sono in alcun spazio dei nomi. **Non** li necessario inserire nel namespace SVG.

Qui mostriamo SVG ospitato all'interno di HTML, poiché si tratta di un caso comune:

```
<!doctype HTML>
<html><title>Creating an Element</title>
<body>
  <svg xmlns="http://www.w3.org/2000/svg"
       width="100%" height="100%"
       viewBox="0 0 400 300"></svg>

  <script>
     var svgNS = "http://www.w3.org/2000/svg";
```

```
    // Create a circle element (not part of the DOM yet)
    var circle = document.createElementNS(svgNS,'circle'); // Creates a <circle/>
    circle.setAttribute('fill','red'); // Note: NOT setAttributeNS()
    circle.setAttribute('cx',150);     // setAttribute turns 150 into a string
    circle.setAttribute('cy','80');    // using a string works, too
    circle.setAttribute('r',35);       // give the circle a radius so we can see it

    // Now, add the circle to the SVG document so we can see it
    var svg = document.querySelector('svg'); // the root <svg> element
    svg.appendChild(circle);
  </script>
</body></html>
```

Ci sono alcuni attributi che devono essere creati in un particolare spazio dei nomi. Sono quelli elencati con i due punti nei loro nomi nell'indice degli attributi SVG . Nello specifico, sono: `xlink:actuate` `xlink:arcrole` , `xlink:arcrole` , `xlink:href` , `xlink:role` , `xlink:show` , `xlink:title` , `xlink:type` , `xml:base` , `xml:lang` e `xml:space` . Imposta questi attributi usando `setAttributeNS()` :

```
var svgNS   = "http://www.w3.org/2000/svg";
var xlinkNS = "http://www.w3.org/1999/xlink";
var img = document.createElementNS( svgNS, 'image' );
img.setAttributeNS( xlinkNS, 'href', 'my.png' );
```

Se crei spesso elementi, in particolare con molti attributi, una funzione di supporto come la seguente può farti risparmiare digitazione, evitare errori e rendere il tuo codice più facile da leggere:

```
<!doctype HTML>
<html><title>Creating an Element</title>
<body>
  <svg xmlns="http://www.w3.org/2000/svg"></svg>
  <script>
     var svg = document.querySelector('svg');
     var circle = createOn( svg, 'circle', {fill:'red', cx:150, cy:80, r:35} );

    // Create an SVG element on another node with a set of attributes.
    // Attributes with colons in the name (e.g. 'xlink:href') will automatically
    // find the appropriate namespace URI from the SVG element.
    // Optionally specify text to create as a child node, for example
    //   createOn(someGroup,'text',{x:100,'text-anchor':'middle'},"Hello World!");
    function createOn(parentEl,name,attrs,text){
      var doc=parentEl.ownerDocument, svg=parentEl;
      while (svg && svg.tagName!='svg') svg=svg.parentNode;
      var el = doc.createElementNS(svg.namespaceURI,name);
      for (var a in attrs){
        if (!attrs.hasOwnProperty(a)) continue;
        var p = a.split(':');
        if (p[1]) el.setAttributeNS(svg.getAttribute('xmlns:'+p[0]),p[1],attrs[a]);
        else      el.setAttribute(a,attrs[a]);
      }
      if (text) el.appendChild(doc.createTextNode(text));
      return parentEl.appendChild(el);
    }
  </script>
</body></html>
```

Lettura / Scrittura attributi

È possibile utilizzare i metodi DOM Level 2 Core `getAttribute()` , `getAttributeNS()` , `setAttribute()` e `setAttributeNS()` per leggere e scrivere valori dagli elementi SVG oppure utilizzare proprietà e metodi personalizzati specificati nell'IDL SVG 1.1 (interfaccia Linguaggio di definizione).

Semplici attributi numerici

Ad esempio, se si dispone di un elemento cerchio SVG:

```
<circle id="circ" cx="10" cy="20" r="15" />
```

puoi utilizzare i metodi DOM per leggere e scrivere gli attributi:

```
var circ = document.querySelector('#circ');
var x = circ.getAttribute('cx') * 1; // Use *1 to convert from string to number value
circ.setAttribute('cy', 25);
```

... oppure è possibile utilizzare le proprietà personalizzate `cx` , `cy` e `r` definite per `SVGCircleElement` . Si noti che questi non sono numeri diretti, ma invece, come con molti accessori in SVG 1.1, consentono l'accesso ai valori animati. Queste proprietà sono di tipo `SVGAnimatedLength` . Ignorando le animazioni e le unità di lunghezza, puoi usare attributi come:

```
var x = circ.cx.baseVal.value; // this is a number, not a string
circ.cy.baseVal.value = 25;
```

trasformazioni

I gruppi SVG possono essere utilizzati per spostare, ruotare, ridimensionare e in altro modo trasformare più elementi grafici nel loro complesso. (Per i dettagli sulle traduzioni SVG, consultare il Capitolo 7). Ecco un gruppo che fa una faccina che è possibile regolare la dimensione, la rotazione e la posizione di regolando la `transform` :

```
<g id="smiley" transform="translate(120,120) scale(5) rotate(30)">
  <circle r="20" fill="yellow" stroke-width="2"/>
  <path fill="none" d="M-10,5 a 5 3 0 0 0 20,0" stroke-width="2"/>
  <circle cx="-6" cy="-5" r="2" fill="#000"/>
  <circle cx="6"  cy="-5" r="2" fill="#000"/>
</g>
```

L'uso dello script per regolare la scala di questo tramite i metodi DOM richiede la manipolazione dell'intero attributo `transform` come stringa:

```
var face = document.querySelector('#smiley');

// Find the full string value of the attribute
var xform = face.getAttribute('transform');
```

```
// Use a Regular Expression to replace the existing scale with 'scale(3)'
xform = xform.replace( /scale\s*\([^)]+\)/, 'scale(3)' );

// Set the attribute to the new string.
face.setAttribute('transform',xform);
```

Con SVG DOM è possibile attraversare le trasformazioni specifiche dell'elenco, trovare quello desiderato e modificare i valori:

```
var face = document.querySelector('#smiley');

// Get the SVGTransformList, ignoring animation
var xforms = face.transform.baseVal;

// Find the scale transform (pretending we don't know its index)
for (var i=0; i<xforms.numberOfItems; ++i){
  // Get this part as an SVGTransform
  var xform = xforms.getItem(i);
  if (xform.type == SVGTransform.SVG_TRANSFORM_SCALE){
    // Set the scale; both X and Y scales are required
    xform.setScale(3,3);
    break;
  }
}
```

- Per attraversare e manipolare il numero di trasformazioni, vedere `SVGTransformList` .
- Per manipolare le singole trasformazioni, vedere `SVGTransform` .

Trascinando elementi SVG

Usare il mouse per trascinare un elemento SVG (o un gruppo di elementi) può essere ottenuto da:

1. L'aggiunta di `mousedown` gestore per inizio la resistenza: l'aggiunta di una traduzione sul elemento da utilizzare durante il trascinamento (se necessario), il monitoraggio `mousemove` eventi, e l'aggiunta di un `mouseup` gestore di porre fine alla resistenza.
2. Durante il `mousemove` , trasforma la posizione del mouse dalle coordinate dello schermo nelle coordinate locali dell'oggetto che stai trascinando e aggiorna la traduzione di conseguenza.
3. Durante il `mouseup` , rimuovendo i gestori di `mousemove` e `mouseup` .

```
// Makes an element in an SVG document draggable.
// Fires custom `dragstart`, `drag`, and `dragend` events on the
// element with the `detail` property of the event carrying XY
// coordinates for the location of the element.
function makeDraggable(el){
  if (!el) return console.error('makeDraggable() needs an element');
  var svg = el;
  while (svg && svg.tagName!='svg') svg=svg.parentNode;
  if (!svg) return console.error(el,'must be inside an SVG wrapper');
  var pt=svg.createSVGPoint(), doc=svg.ownerDocument;

  var root = doc.rootElement || doc.body || svg;
  var xlate, txStartX, txStartY, mouseStart;
  var xforms = el.transform.baseVal;

  el.addEventListener('mousedown',startMove,false);
```

```
  function startMove(evt){
    // We listen for mousemove/up on the root-most
    // element in case the mouse is not over el.
    root.addEventListener('mousemove',handleMove,false);
    root.addEventListener('mouseup',  finishMove,false);

    // Ensure that the first transform is a translate()
    xlate = xforms.numberOfItems>0 && xforms.getItem(0);
    if (!xlate || xlate.type != SVGTransform.SVG_TRANSFORM_TRANSLATE){
      xlate = xforms.createSVGTransformFromMatrix( svg.createSVGMatrix() );
      xforms.insertItemBefore( xlate, 0 );
    }
    txStartX=xlate.matrix.e;
    txStartY=xlate.matrix.f;
    mouseStart = inElementSpace(evt);
    fireEvent('dragstart');
  }

  function handleMove(evt){
    var point = inElementSpace(evt);
    xlate.setTranslate(
      txStartX + point.x - mouseStart.x,
      txStartY + point.y - mouseStart.y
    );
    fireEvent('drag');
  }

  function finishMove(evt){
    root.removeEventListener('mousemove',handleMove,false);
    root.removeEventListener('mouseup',  finishMove,false);
    fireEvent('dragend');
  }

  function fireEvent(eventName){
    var event = new Event(eventName);
    event.detail = { x:xlate.matrix.e, y:xlate.matrix.f };
    return el.dispatchEvent(event);
  }

  // Convert mouse position from screen space to coordinates of el
  function inElementSpace(evt){
    pt.x=evt.clientX; pt.y=evt.clientY;
    return pt.matrixTransform(el.parentNode.getScreenCTM().inverse());
  }
}
```

Leggi Scripting online: https://riptutorial.com/it/svg/topic/5021/scripting

Capitolo 21: Sfumature

Parametri

Comune	definizione
gradientUnits	il sistema di coordinate degli attributi del gradiente. O objectBoundingBox o userSpaceOnUse
gradientTransform	la trasformazione da applicare ai contenuti del gradiente
spreadMethod	definisce cosa succede al di fuori dei confini del gradiente. O pad, riflettere o ripetere
xlink: href	link a un altro gradiente che fornisce attributi o contenuti
------	------
Gradiente lineare	Definizione
------	------
x1	definisce il vettore del gradiente
x2	vedi x1
Y1	vedi x1
y2	vedi x1
------	------
Gradiente radiale	Definizione
------	------
cx	la coordinata x del centro del gradiente esterno
cy	la coordinata y del centro del gradiente esterno
r	il raggio esterno del gradiente. La posizione di una fermata al 100%
fx	la coordinata x del centro del gradiente interno. La posizione di una fermata dello 0%
fy	la posizione y del centro del gradiente interno. La posizione di una fermata dello 0%

Osservazioni

SVG fa distinzione tra maiuscole e minuscole quindi ricorda di usare una G maiuscola nel mezzo.

Examples

linearGradient

```
<svg>
  <defs>
        <linearGradient id='g' y1="100%" x2="100%">
            <stop offset='0%' stop-color='yellow' />
            <stop offset='100%' stop-color='green' />
        </linearGradient>
    </defs>
    <rect width='100%' height='100%' fill='url(#g)'/>
</svg>
```

radialGradient

```
<svg>
  <defs>
    <radialGradient id="g">
      <stop offset="10%" stop-color="green" />
      <stop offset="90%" stop-color="white" />
    </radialGradient>
  </defs>

  <rect width='100%' height='100%' fill='url(#g)'/>
</svg>
```

Leggi Sfumature online: https://riptutorial.com/it/svg/topic/3346/sfumature

Capitolo 22: Testo

Parametri

<text>	Dettagli
X	La posizione x del testo.
y	La posizione y del testo.
dx	Spostamento relativo in posizione x.
dy	Spostamento relativo in posizione y.
ruotare	Specifica lo spostamento angolare per glifi di testo.
textLength	Adatta il testo alla lunghezza specificata.
lengthAdjust	Specifica se solo crenatura o crenatura e glifi vengono compressi / stirati per adattare il testo alla lunghezza del testo specificata. Valori: spaziatura o spaziaturaAndGlyphs
-	**Parametri comuni a tutti gli elementi di suddivisione del testo (testo, tref, textPath, tspan)**
text-anchor	Specifica l'allineamento orizzontale. Valori: inizio, metà, fine.
basale-shift	Sposta la baseline del testo in base a entrambi i valori forniti dalla tabella font per il posizionamento in apice o pedice (sub, super) o in percentuale o lunghezza positiva o negativa. Valori: sub, super,% o lunghezza.

Osservazioni

baseline-shift non è supportato dalle versioni più recenti dei browser Firefox e Microsoft a luglio 2016.

Examples

Disegna il testo

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text x="40" y="60" font-size="28">Hello World!</text>
</svg>
```

La coordinata xey specifica la posizione dell'angolo in basso a sinistra del testo (a meno che

l'ancoraggio del testo non sia stato modificato).

Super e pedice

Utilizzando il parametro spostamento di linea di base, è possibile specificare super- o pedice. Ma questo non è supportato da tutti i principali browser.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text x="10" y="20">x<tspan baseline-shift="super">2</tspan></text>
    <text x="10" y="60">f<tspan baseline-shift="sub">x</tspan></text>
</svg>
```

Per una soluzione cross-browser, è possibile utilizzare dy, dx e la dimensione del carattere relativa.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text x="10" y="40">x<tspan dy="-7" font-size=".7em">2</tspan></text>
    <text x="10" y="80">f<tspan dy="3" font-size=".7em">x</tspan></text>
</svg>
```

Ruota il testo

La proprietà rotate ruota ciascun carattere per l'angolo specificato.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text x="10" y="20" rotate="30">Each character is rotated</text>
</svg>
```

Per ruotare l'intero elemento di testo, devi usare la proprietà transform.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <text transform="translate(10, 60) rotate(30)">The whole text is rotated</text>
</svg>
```

Posizionamento di singole lettere con matrici di valori X e Y.

```
<svg width="400px" height="200px">
 <text x="1em, 2em, 3em, 4em, 5em" y="3em, 4em, 5em">
    Individually Spaced Text
  </text>
</svg>
```

L'elemento Testo supporta il posizionamento individuale di lettere accettando una matrice di valori per x e y.

Leggi Testo online: https://riptutorial.com/it/svg/topic/3033/testo

Capitolo 23: Trasformazione

Osservazioni

Gli elementi grafici possono essere modificati aggiungendo un attributo *transform* .

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="0" width="30" height="30" transform="translate(10, 10)" />
</svg>
```

Invece che l'origine in alto a sinistra viene renderizzata alle coordinate (0, 0), verrà mostrata in basso ea destra, dal punto (10, 10).

Intere gruppi di elementi possono essere trasformati insieme e le trasformazioni possono aggiungersi l'una all'altra.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <g transform="rotate(45)">
        <rect x="0" y="0" width="30" height="30" />
        <circle cx="5" cy="5" r="5" transform="scale(3)" />
    </g>
</svg>
```

Innanzitutto, ogni punto del cerchio verrà ridimensionato con il fattore 3 in relazione all'origine, portando il centro del cerchio al centro del rettangolo in (15, 15) e il suo raggio a 15. Quindi, il rettangolo e il il cerchio verrà ruotato di 45 gradi in senso orario attorno all'origine.

Examples

tradurre

Sposta un rettangolo di 10 unità a destra e 20 unità in basso:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="0" width="30" height="30" transform="translate(10, 20)" />
</svg>
```

Sposta 0 unità orizzontalmente e 20 unità in alto:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="20" width="30" height="30" transform="translate(0, -20)" />
</svg>
```

Sposta 10 unità a sinistra e 0 unità verticalmente:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="10" y="0" width="30" height="30" transform="translate(-10)" />
```

```
</svg>
```

scala

Ridimensionare un rettangolo orizzontalmente per il fattore 2 e verticalmente per il fattore 0.5:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="10" y="10" width="40" height="40" transform="scale(2, 0.5)" />
</svg>
```

Il risultato è equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="20" y="5" width="80" height="20" />
</svg>
```

Specchia un rettangolo in orizzontale:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="0" width="20" height="40" transform="scale(-1, 1)" />
</svg>
```

La scala funziona in relazione all'origine, quindi questo è equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="-20" y="0" width="20" height="40" />
</svg>
```

ruotare

Ruota un poligono in senso orario di 90 gradi attorno all'origine:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 15,20" transform="rotate(90)" />
</svg>
```

Il risultato è equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 0,30 -20,15" />
</svg>
```

Il centro di rotazione può essere dato esplicitamente:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 15,20" transform="rotate(90, 15, 15)" />
</svg>
```

Il risultato è equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="30,0 30,30 10,15" />
</svg>
```

skewX, skewY

inclinare un poligono in orizzontale di 45 gradi:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 30,30 0,30" transform="skewX(45)" />
</svg>
```

Il risultato è equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 60,30 30,30" />
</svg>
```

i valori x sono calcolati come `x + y * tan(angle)` , i valori y rimangono invariati

inclina verticalmente un poligono di 30 gradi:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 30,30 0,30" transform="skewY(30)" />
</svg>
```

Il risultato è equivalente a (tenendo conto dell'arrotondamento)

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,17.32 30,47.32 0,30" />
</svg>
```

i valori x rimangono invariati, i valori y sono calcolati come `y + x * tan(angle)`

matrice

Applicare una matrice di trasformazione a un poligono:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="0,0 30,0 30,30 0,30" transform="matrix(1,0.6,-1.2,1,40,10)" />
</svg>
```

Ogni punto (x, y) verrà trasformato applicando la matrice (a, b, c, d, e, f) in questo modo:

```
┌ x_new ┐     ┌ a  c  e ┐ * ┌ x_old ┐    ┌ x_old * a + y_old * c + e ┐
└ y_new ┘ =   └ b  d  f ┘   │ y_old │ =  └ x_old * b + y_old * d + f ┘
                            └   1   ┘
```

Il risultato è equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <polygon points="40,10 70,28 34,58 4,40" />
</svg>
```

Trasformazioni multiple

Le trasformazioni possono essere concatenate e applicate da **destra a sinistra**

Ruota un rettangolo di 90 gradi, quindi spostalo verso il basso di 20 unità e verso destra di 20 unità:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="-10" y="-20" width="20" height="40"
        transform="translate(20 20) rotate(90)" />
</svg>
```

Il risultato è equivalente a

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="0" y="10" width="40" height="20" />
</svg>
```

Leggi Trasformazione online: https://riptutorial.com/it/svg/topic/3249/trasformazione

Capitolo 24: Trasformazione

Sintassi

- transform = " *[funzioni]* * "
- translate (x [, y])
- ruotare (θ [, x, y])
- scala (x [, y])
- skewX (θ)
- skewY (θ)
- matrice (a, b, c, d, e, f)

Examples

Applicazione delle trasformazioni

Le trasformazioni possono essere applicate agli elementi aggiungendo un attributo di `transform` :

```
<circle cx="0" cy="0" r="50" transform="translate(50,50)"/>
```

O a gruppi di elementi racchiusi tra tag `<g>` :

```
<g transform="translate(50,50)">
<circle cx="0" cy="0" r="50"/>
<circle cx="0" cy="0" r="25" fill="white"/>
</g>
```

Più trasformazioni possono essere applicate allo stesso elemento aggiungendole separate da spazi:

```
<circle cx="0" cy="0" r="50" transform="translate(50,50) scale(.5)"/>
```

Funzioni di trasformazione

Tradurre

`translate` sposta le immagini lungo i vettori specificati:

```
<circle cx="0" cy="0" r="50" transform="translate(50,50)"/>
```

Il primo valore è la traduzione xe il secondo y. Se y viene omesso, verrà impostato su 0.

Scala

`scale` ridimensiona elementi di rapporti specificati:

```
<circle cx="50" cy="50" r="25" transform="scale(.5,2)"/>
```

Come `translate` , gli argomenti sono x, quindi y. Tuttavia, in `scale` , se y viene omesso, verrà impostato di default sul valore di x; in altre parole, ridimensiona l'elemento senza modificare le proporzioni.

Ruotare

`rotate` ruota gli elementi per angoli specificati.

```
<!-- <rect> used for this example because circles can't be rotated -->
<rect width="100" height="5" transform="rotate(90,50,50)"/>
```

Il primo valore è l'angolo, in gradi. La trasformazione viene applicata in senso orario. Gli altri due valori rappresentano il punto da ruotare attorno, in modo predefinito all'origine.

Leggi Trasformazione online: https://riptutorial.com/it/svg/topic/7100/trasformazione

Capitolo 25: uso

Parametri

Parametro	Dettagli
X	coordinata dell'asse x dell'angolo in alto a sinistra
y	coordinata dell'asse y dell'angolo in alto a sinistra
larghezza	larghezza dell'elemento `<use>`
altezza	altezza dell'elemento `<use>`
xlink: href	identificativo risorsa (si riferisce all'ID di un altro elemento) SVG 2 propone di deprecare questo e sostituirlo con un semplice attributo href

Osservazioni

I dettagli possono essere trovati nella Raccomandazione W3C per SVG e nella nuova Raccomandazione Candidata per SVG2

Examples

Usando un'icona

L'elemento `<use>` viene spesso utilizzato per icone riutilizzabili, in collaborazione con l'elemento `<symbol>` . Questo sembra:

```
<svg>
    <symbol viewBox="0 0 16 16" id="icon-star">
        <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.021l4.912 3.57-1.904
5.834h.021l4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
    </symbol>
</svg>
```

E l'elemento `<use>` :

```
<svg>
    <use xlink:href="#icon-star"/>
</svg>
```

L'elemento `<use>` copia il `<symbol>` e lo visualizza. Puoi anche sovrascrivere gli stili sul `<symbol>` sui singoli elementi `<use>` , ad es

```
<style>
```

```
    .red {
        fill: red;
    }
</style>

<svg>
    <use class="red" xlink:href="#icon-star"/>
</svg>
```

Leggi uso online: https://riptutorial.com/it/svg/topic/6904/uso

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con SVG	almcd, Community, Robert Longson, Timothy Miller, uruloke, w5m, web-tiki
2	Animazione	Joachim Schirrmacher, Michael Mullany
3	Cerchio	almcd, Kake_Fisk, w5m
4	clipPath	Danny_ds, lodz
5	Colori	Danny_ds, Holger Will, Joachim Schirrmacher, Michael Mullany, w5m
6	Creazione di caratteri	Holger Will
7	defs	Michael Mullany
8	Ellisse	adius
9	filtri	Anko, Michael Mullany, RamenChef
10	interruttore	lodz
11	L'elemento SVG	Michael Mullany, Timothy Miller
12	Linea	Deni Spasovski, Michael Mullany, w5m
13	marcatore	Michael Mullany
14	maschera	Holger Will
15	Patterns	Michael Mullany, Robert Longson
16	percorsi	Malcolm McLean, Michael Mullany, w5m
17	pointer-events	Holger Will
18	polilinea	adius, Michael Mullany
19	Rettangolo	almcd, w5m
20	Scripting	Michael Mullany, Phrogz
21	Sfumature	Robert Longson

22	Testo	Kake_Fisk, Michael Mullany
23	Trasformazione	ccprog, Michael Mullany, Stephen Leppik
24	uso	Robert Longson, Timothy Miller