DHTML, JavaScript
Il disegno di linee, Cerchio, Ellisse (ovale), Polyline, Polygon, Rettangolo.
High Performance JavaScript Vector Graphics Library.

Sviluppato da Walter Zorn

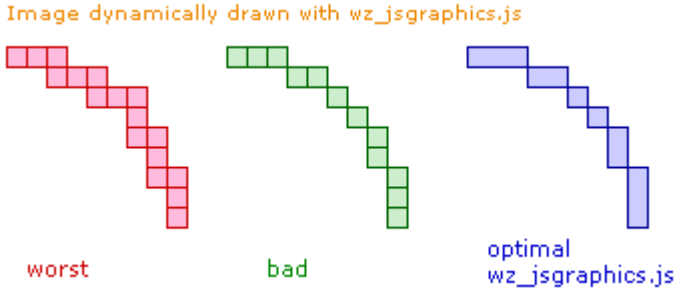
Questo VectorGraphics è una libreria JavaScript che fornisce capacità grafiche per Javascript: funzioni per disegnare cerchi, ellissi (ovali), linee oblique, polilinee e poligoni (ad esempio triangoli, rettangoli) dinamicamente in una pagina web. L'uso di questa libreria grafica vettoriale dovrebbe essere facile anche se non si ha esperienza in JavaScript. **Documentazione** . Un altro obiettivo durante lo sviluppo di questo prodotto era ottenere **prestazioni** ottimizzate.

Ecco un esempio di **come animare (ruotare) una vectorgraphic** .

Vedere anche **come disegnare su un'immagine** , utilizzando l'immagine stessa come tela (in questo esempio viene utilizzato **DHTML-Drag & Drop-Library**).

Performance

In HTML ci sono elementi come linee oblique, cerchi, ellissi o altre aree non-rettangolari. Per una soluzione, i pixel possono essere dipinti con la creazione di piccoli strati di colore di sfondo (elementi DIV), e l'organizzazione di questi si può adattare al modello desiderato. Una creazione di un DIV separato per ogni pixel del modello, tuttavia, sarebbe molto inefficiente.



Per ridurre al minimo la quantità di overhead e per ottimizzare le prestazioni per quanto è possibile, questo prodotto, oltre all'utilizzo di algoritmi veloci (basati su algoritmi di Bresenham) per calcolare le forme, riduce al minimo il numero di DIV. Pertanto, come illustrato dalla foto a sinistra, il numero di elementi DIV è minima.

Tuttavia, non bisogna confrontare le prestazioni con Java o applicazioni stand-alone!
L'interpretazione del browser di elementi DIV è ovviamente molto più lenta di una semplice colorazione di pixel. Anche questa libreria non può sottrarsi da tale restrizione fondamentale - si cerca solo di ottenere il meglio possibile. Si raccomanda di astenersi dalla creazione di forme che si estendono oltre 600 - 1000 pixel in entrambe le dimensioni.

Alternativa a SVG o <canvas> tag?
Entrambe le tecniche sono attualmente non supportate nativamente da IE. IE è proprietaria di VML che, d'altra parte, non è supportato dagli altri browser. Pertanto è possibile utilizzare questa Vector Graphics Library come alternativa presumibilmente in esecuzione su macchine di oltre il 95% di tutti i visitatori di Internet.

Cross funzionalità del browser?

Linux:
Browser con motore Gecko (Mozilla, Netscape 6 +, Galeon), Konqueror, Opera 5, 6 e 7 +.

Windows:
Gecko-Browser, IE 4, 5 e 6, Opera 5, 6 e 7 +.

Mac:
Safari, Gecko-Browser, Opera, parzialmente IE.

La funzionalità "Disegna in elementi HTML anche dopo che la pagina è stata caricata completamente" non è disponibile per lOpera prima della versione 7, mentre "Draw nel documento mentre la pagina viene analizzata" è cross-browser.

Download

wz_jsgraphics.js 3.05, pubblicato sotto licenza **LGPL:** **wz_jsgraphics.zip** (7 KB)

Storia degli aggiornamenti

2. Inserire la libreria

Decomprimere il file nella stessa directory del file HTML. Inserire il seguente codice nella sezione intestazione del file html, tra <head> e </ head> - se si desidera avere la libreria in un altra directory, adattare il percorso src = "wz_jsgraphics.js":

```
<script type="text/javascript" src="wz_jsgraphics.js"> </ script>
```

3. HTML: elementi come tele

Questo passaggio non è richiesto per la modalità "Disegna direttamente nel documento mentre la pagina viene analizzata". Serve per disegnare in elementi DIV anche dopo che la pagina è stata caricata completamente, posizionando (in modo relativo o assoluto) strati (elementi DIV) che sono usati come tele. Ognuno di questi livelli deve avere un ID univoco:

 </ div>
...
 </ div>

Come disegnare forme

1. Creazione di un oggetto jsGraphics

a) Disegnare in elementi DIV anche dopo che la pagina è completamente caricata:
(Questa modalità non funziona con Opera 5 e 6.)
Si veda l'esempio qui sotto per come creare un oggetto per jsGraphics in un DIV cioè un elemento destinato a servire come tela. Il codice deve essere inserito nel codice html all'interno dell'elemento DIV in questione ed in ogni caso prima del tag di chiusura del body. Si crea un oggetto grafico con l'operatore new e la funzione costruttore `jsGraphics ()` che richiede come parametro o l'ID del DIV che costituisce la tela tra virgolette, o un riferimento diretto a tale DIV:

```
<script type="text/javascript">
<!--
var jg = new jsGraphics("myCanvas");
//-->
</script>
```

or

```
<script type="text/javascript">
<!--
var cnv = document.getElementById("myCanvas");
var jg = new jsGraphics(cnv);
//-->
</script>
```

Se ci sono più elementi separati DIV, ognuno richiede una propria istanza di tipo jsGraphics:

```
<script type="text/javascript">
<!--

var jg = new jsGraphics("myCanvas");
var jg2 = new jsGraphics("anotherCanvas");

//-->
</script>
```

b) Disegnare direttamente nel documento mentre la pagina viene analizzata:
Funziona anche con Opera 5 / 6. Anziché l'ID di un elemento DIV non deve essere passato alla funzione costruttore nessun parametro (o valore null):

```
<script type="text/javascript">
<!--

var jg doc = new jsGraphics();

//-->
</script>
```

Invece di jg, jg2 o jg_doc si può scegliere qualsiasi altro nome di variabile, a condizione che si seguano le regole per i nomi delle variabili in JavaScript.

2. Funzioni per disegnare forme

Una volta generati, questi oggetti grafici (in questo esempio jg, jg2 e / o jg_doc) possono essere utilizzati per chiamare i metodi per disegnare le forme. Le forme generate da un certo oggetto grafico saranno disegnate nell'elemento html in questione, o, con modalità b) utilizzate, nel documento stesso:

```
<script type="text/javascript">
<! -
function myDrawFunction ()
(
jg_doc.setColor ("# 00FF00"); / / verde
```

```
jg_doc.fillEllipse (100, 200, 100, 180); / / coordinate relative al documento
jg_doc.setColor ("marrone");
jg_doc.drawPolyline (new Array (50, 10, 120), new Array (10, 50, 70));
jg_doc.paint (); / / disegna, in questo caso, direttamente nel documento

jg.setColor ("# ff0000"); / / rosso
jg.drawLine (10, 113, 220, 55); / / coordinate correlate a "MyCanvas"
jg.setColor ("# 0000FF"); / / blu
jg.fillRect (110, 120, 30, 60);
jg.paint ();

jg2.setColor ("# 0000FF"); / / blu
jg2.drawEllipse (10, 50, 30, 100);
jg2.drawRect (400, 10, 100, 50);
jg2.paint ();
)

var jg_doc = new jsGraphics(); / / disegna direttamente nel documento
var jg = new jsGraphics ("MyCanvas");
var jg2 = new jsGraphics ("anotherCanvas");

myDrawFunction ();

//-->
</ Script>
```

Come illustrato in questo esempio, in un primo momento il colore della "penna" dovrebbe essere impostato. In caso contrario, sarà utilizzato il colore predefinito nero. Le coordinate passate ai metodi per il disegno delle forme fanno riferimento, se la modalità è b) al punto in alto a sinistra del documento stesso, altrimenti al punto in alto a sinistra del DIV che costituisce la tela. Per ogni tela (oggetto grafico) separatamente, deve essere chiamato esplicitamente il proprio metodo paint () per visualizzare sullo schermo la grafica generata nella pagina html.

Nome della funzione (metodo)	**esempio di codice** da poter essere utilizzato con l'oggetto del precedente esempio 'jg' (che è il contesto grafico del DIV chiamato "myCanvas")

Note Generali

1.) Numeri passati a queste funzioni devono essere sempre **numeri interi,** non numeri decimali (numeri in virgola mobile), né caratteri o stringhe. Per esempio, i valori ottenuti dai formulari sono sempre stringhe e i risultati ottenuti da precedenti calcoli possono essere in virgola mobile. Utilizzare allora le funzioni di conversione di JavaScript `parseInt()` o `Math.round()` per convertire tali valori in numeri interi.

Esempio: `jg.setStroke(parseInt(document.MyForm.Linewidth.value));`

2.) Si consideri che le coordinate si trovano tra i pixel, non su di essi, e che il disegno della "penna" si blocca al di sotto e a destra del percorso specificato dalle coordinate passate alle funzioni.

`setColor(` `"#HexColor"` **`);`**
 	
Specifica il colore del disegno a "penna". Una volta impostato, questo colore sarà utilizzato dalla chiamata dei metodi di disegno fino a quando non verrà sovrascritto da un'altra chiamata di setColor (). Il valore deve essere racchiuso tra virgolette, e dovrebbe essere esadecimale seguendo il modello "# RRGGBB" (come al solito con l'HTML). Possono essere usati anche I nomi dei colori disponibili in formato HTML (ad esempio " maroon")
	`jg.setColor("#ff0000");`
	
	o con risultato identico
	
	`jg.setColor("red");`

`setStroke(` `Number` **`);`**
 	
Specifica lo spessore del disegno a "penna" per le linee e le linee di delimitazione delle forme. Una volta impostato, questo spessore sarà utilizzato dalla chiamata dei metodi di disegno fino a quando non verrà sovrascritto da un'altra chiamata di setStroke (). Spessore della linea di default è di 1 px , fino a quando setStroke () non viene invocato.
Per creare linee tratteggiate, `setStroke ()` richiede la costante `Stroke.DOTTED` come argomento. La larghezza di linee tratteggiate è sempre di 1 pixel.
	`jg.setStroke(3);`
	
	o
	
	`jg.setStroke(Stroke.DOTTED);`

`drawLine(` `X1, Y1, X2, Y2` **`);`** Linea dalla prima alla seconda coppia di coordinate. Spessore della linea o è 1 px o il valore più recentemente specificato da **`.setStroke()`**	`jg.drawLine(20,50,453,40);`

`drawPolyline( Xpoints, Ypoints );`

 Una polilinea è una serie di segmenti collegati. Xpoints e Ypoints sono matrici che specificano le coordinate x e y di ciascun punto come segue:

```
var Xpoints = new Array(x1,x2,x3,x4,x5);
var YPoints = new Array(y1,y2,y3,y4,y5);
```

Invece di Xpoints e Ypoints si possono naturalmente utilizzare altri nomi a condizione che questi seguano le regole per i nomi di variabili JavaScript.

lo spessore della linea o è 1px o il valore più recentemente specificato da **`.setStroke()`**

```
var Xpoints = new Array(10,85,93,60);
var YPoints = new Array(50,10,105,87);
jg.drawPolyline(Xpoints,Ypoints);
```

`drawRect( X, Y, width, height );`

 Schema di un rettangolo . X e Y sono le coordinate del punto in alto a sinistra. Spessore della linea o è 1px o il valore specificato dal più recente **`.setStroke()`**

```
jg.drawRect(20,50,70,140);
```

`fillRect( X, Y, width, height );`

 rettangolo pieno. X e Y sono le coordinate dell'angolo in alto a sinistra.

```
jg.fillRect(20,50,453,40);
```

`drawPolygon( Xpoints, Ypoints );`

 Poligono. Xpoints e Ypoints sono matrici che specificano le coordinate x e y degli angoli del poligono come segue:

```
var Xpoints = new Array(x1,x2,x3,x4,x5);
var Ypoints = new Array(y1,y2,y3,y4,y5);
```

Il poligono verrà chiuso automaticamente se il primo e l'ultimo punto non sono identici.

lo spessore della linea o è 1px o il valore più recentemente specificato da **`.setStroke()`**

```
var Xpoints = new Array(10,85,93,60);
var Ypoints = new Array(50,10,105,87);
jg.drawPolygon(Xpoints, Ypoints);
```

Invece di Xpoints e Ypoints si possono naturalmente utilizzare altri nomi a condizione che questi seguano le regole per i nomi delle variabili.

`fillPolygon( Xpoints, Ypoints );`

 Poligono riempito. Parametri come per drawPolygon ()

```
jg.fillPolygon(new Array(10,85,93,60),
new Array(50,10,105,87));
```

`drawEllipse( X, Y, width, height );`

 Schema di un'ellisse. I valori si riferiscono al rettangolo di delimitazione dell'ellisse, X e Y sono le coordinate dell'angolo superiore sinistro del rettangolo, piuttosto che del suo centro.

Spessore della linea o è 1px o il valore specificato dal più recente **`.setStroke()`**

```
jg.drawEllipse(20,50,70,140);
o
jg.drawOval(20,50,70,140);
```

`fillEllipse( X, Y, width, height );`

 Un'ellisse piena. I valori fanno riferimento al rettangolo di delimitazione dell'ellisse, X e Y sono le coordinate dell'angolo in alto a sinistra del rettangolo, piuttosto che del suo centro.

```
jg.fillEllipse(20,50,71,141);
o
jg.fillOval(20,50,71,141);
```

`fillArc( X, Y, width, height, start-angle, end-angle );`

 Riempie una sezione di torta di un'ellisse. Start-angolo e end-angolo possono essere numeri interi o valori decimali. Come con le altre funzioni
Ellipse () , X e Y specificano l'angolo in alto a sinistra del rettangolo di delimitazione.

```
jg.fillArc(20,20,41,12,270.0,220.0);
```

`setFont( "font-family", "size+unit", Style );`

Questo metodo può essere invocate prima di drawString () per specificare o cambiare il font-family, le dimensioni e lo stile. Valori o font-family e dimensioni possono essere quelli disponibili in HTML, e devono essere racchiusi tra virgolette.

Stili di carattere disponibili:

Font.PLAIN per lo stile normale (non grassetto, non corsivo)
Font.Bold per i caratteri in grassetto
Font.ITALIC per il corsivo
Font.ITALIC_BOLD or Font.BOLD_ITALIC per combinare questi ultimi.

Esempio: vedi `drawString()` sotto

drawString("Text", X, Y);

I've been drawn with the Vectorgraphics Library...

Scrive il testo alla posizione specificata da X e Y. A differenza di Java, queste coordinate si riferiscono al punto in alto a sinistra della prima riga del testo. La stringa passata a drawString () deve essere racchiusa tra virgolette. (Non-escape) I tag HTML all'interno della stringa vengono interpretati. Ad esempio, "Testo
 altro testo " serve a creare un'interruzione di riga.

```
jg.setFont("arial","15px",Font.ITALIC_BOLD);
jg.drawString("Some Text",20,50);
```

drawStringRect("Text", X, Y, Width, Alignment);

A text drawn by using drawStringRect() which allows to set the width of the text rectangle and to align the text horizontally (in this case "right")

Come **drawString**. Consente però di impostare la larghezza del rettangolo di testo e per specificare allineamento orizzontale del testo. L'allineamento del testo deve essere una stringa (cioè racchiusa tra virgolette o apostrofi) e può essere "left", "center", "right" o "justify

```
jg.setFont("verdana","11px",Font.BOLD);
jg.drawStringRect("Text",20,50,300,"right");
```

drawImage("src", X, Y, width, height);

Richiama l'immagine nella posizione indicata. Il parametro "src" specifica il percorso del file. La larghezza e l'altezza sono parametri opzionali (possono essere 0, null o omessi, nel qual caso viene visualizzata l'immagine nella sua dimensione di default), ma offrono la possibilità di allungare l'immagine (quasi) arbitrariamente.

Facoltativamente, drawImage () accetta un quinto parametro, che è possibile utilizzare per inserire un ***EventHandler*** nel tag di immagine generata.

Esempio:

```
jg.drawImage('anImg.jpg',8,5,95,70,'onMouseOver="YourFunc()"');
```

```
jg.drawImage("friendlyDog.jpg",20,50,100,150);
```

paint();

Deve essere invocato espressamente per disegnare la grafica generata internamente nella pagina html. Per ottimizzare le prestazioni si consiglia di trattenersi dal chiamare `paint()` inutilmente solo per brevi intervalli.

Evitare di qualcosa di simile:

```
jg.drawEllipse (0, 0, 100, 100);
jg.paint ();
jg.drawLine (200, 10, 400, 40);
jg.paint ();
...
```

Il seguente codice sarà più veloce:

```
jg.drawEllipse (0, 0, 100, 100);
jg.drawLine (200, 10, 400, 40);
                    /*... altri metodi di disegno ... * /
jg.paint ();
```

```
jg.paint();
```

clear();

Qualsiasi contenuto creato dal JavaScript Graphics Library sarà cancellato (all'interno della tela a cui l'oggetto Graphics si riferisce). Il contenuto predefinito della tela (il contenuto non è creato dallo script) resterà intatto, cioè non sarà cambiato né eliminato.

```
jg.clear();
```

Tutta quello che lo script ha richiamato in "MyCanvas" è soppresso (assumendo che "MyCanvas" sia il DIV per il quale 'jg' è stato creato).

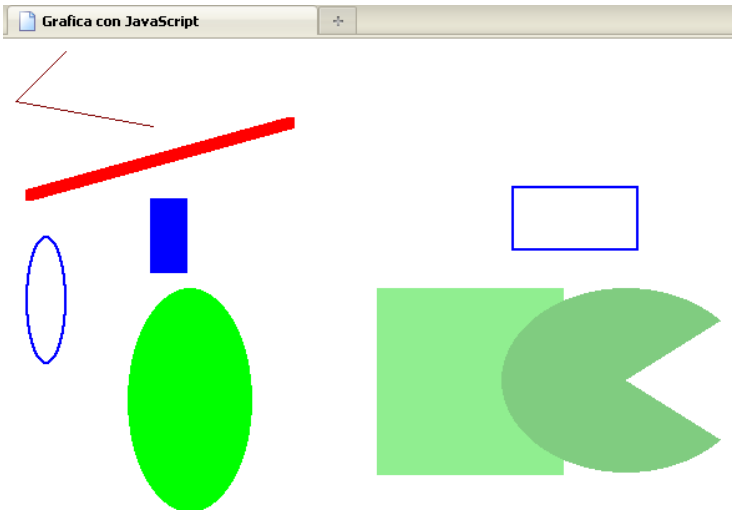
setPrintable(true);

Per impostazione predefinita, la stampa delle forme non è fattibile dato che le impostazioni predefinite di stampa del browser di solito evitano che i colori di sfondo siano stampati. Invocare `setPrintable ()` con il parametro true permette di disegnare forme stampabili (almeno in Mozilla / Netscape e IE 6 +). Tuttavia, al prezzo di una lieve diminuzione della velocità di rendering (circa il 10% al 25% più lento).

```
jg.setPrintable(false);
```

Il parametro false forza alla modalità non-printable. Il vantaggio di questo scelta è la ri-ottimizzazione delle prestazioni del rendering.

Codice Esempio




```
<html>
<head>
<title> Grafica con JavaScript </title>
<meta name="Generator" content="Alleycode HTML Editor">
<script type="text/javascript"
          src="wz_jsgraphics.js"></script>
</head>

<body>

<script type="text/javascript">
<!--
  var jg_doc = new jsGraphics();    // per disegnare direttamente
                                    // nel documento

function disegna()
{
  jg_doc.setColor("#00ff00"); // green
  jg_doc.fillEllipse(100, 200, 100, 180); // coordinate relative
                                    // al documento    ..... ellisse riempita di colore verde  acceso
  jg_doc.setColor("maroon");
  jg_doc.drawPolyline(new Array(50, 10, 120), new Array(10, 50, 70));     // poligonale aperta marrone
  jg_doc.paint();
}
disegna();       // si disegna direttamente nel documento
//-->
</script>

<div id="test"
       style="position:absolute;left:300;top:200;height:150px;width:150px;background-color: lightgreen ">
<script type="text/javascript">
<!--
jg = new jsGraphics("test");                // per disegnare nella tela “test”
jg.setColor('#80cc80');
jg.fillArc(100, 0, 200, 148, 40, 320);    // effetto torta verde senza una fetta … spostata a destra e in basso
jg.paint();                                 // si noti come il disegno esca dalla tela con sfondo colorato di verde chiaro con CSS
//-->
</script>

</div>

<div id="primo"
      style="position:relative;height:100px;width:300%">
<script type="text/javascript">
<!--
  jg1 = new jsGraphics("primo");            //  per disegnare nella tela "primo"
  jg1.setStroke(10);                        //  per uno spessore di 10 px ….. default 1 px
  jg1.setColor("#ff0000");         // red
  jg1.drawLine(10, 113, 220, 55);          // coordinate relative alla tela "primo" … linea rossa spessa 10 px
  jg1.setColor("#0000ff");         // blue
  jg1.fillRect(110, 120, 30, 60);             // rettangolo riempito di colore blu
  jg1.paint();

//-->
</script>
</div>

<div id="secondo"
      style="position:relative;height:100px;width:300px">
<script type="text/javascript">
<!--
  jg2 = new jsGraphics("secondo");          // per disegnare nella tela “secondo”
  jg2.setStroke(2);                         //  per uno spessore  di 2px
  jg2.setColor("#0000ff");            // blue
  jg2.drawEllipse(10, 50, 30, 100);          // ellisse
  jg2.drawRect(400, 10, 100, 50);           // rettangolo  (perimetro blu)
  jg2.paint();
//-->
</script>
</div>

</body>
</html>
```