4D SVG

Introduction

Composant 4D SVG

Le SVG (Scalable Vector Graphics) est un format de fichier graphique vectoriel bidimensionnel basé sur le XML. 4D inclut un moteur de rendu intégré permettant d'afficher les fichiers SVG.

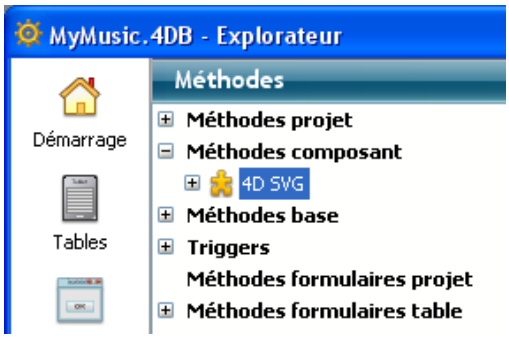
Le langage XML dédié à la manipulation des images SVG est particulièrement riche et étendu. Afin d'en simplifier l'accès et la prise en main, 4D propose le composant 4D SVG. Ce composant comporte de nombreuses commandes permettant la création et la manipulation d'objets graphiques usuels. Le but de cette bibliothèque n'est pas d'être exhaustif mais de répondre aux besoins les plus courants des développeurs 4D. A noter que tous les besoins spécifiques supplémentaires pourront être traités avec les commandes XML de 4D.

Installation et mise en oeuvre

Le Composant 4D SVG doit être installé au minimum dans 4D v11 SQL release 3 (version 11.3). La base hôte doit fonctionner en mode Unicode (le composant ne peut pas être utilisé dans les bases fonctionnant en mode Compatibilité ASCII).

Comme tout composant 4D, le Composant 4D SVG s'installe par la copie du dossier du composant (4D SVG.4dbase) dans le dossier Components de la base. Le dossier Components de la base doit être situé au même niveau que le fichier de structure. Les composants étant chargés au démarrage, la base ne doit pas être lancée avant la copie complète de tous les éléments.

Si le composant est correctement installé, l'élément **4D SVG** apparaît dans la page Méthodes de la base, rubriques "Méthodes composant" :



Vous pouvez déployer cet élément afin de visualiser l'ensemble des commandes du composant. Les commandes s'utilisent dans l'éditeur de méthodes de 4D comme des commandes 4D ou de plug-in standard.

A noter que le Composant 4D SVG vous permet de bénéficier de fenêtres supplémentaires pour la sélection des commandes et le rendu du code SVG. Pour plus d'informations, reportez-vous à la section **Outils de développement**.

Effets SVG et Direct2D (Windows)

A partir de 4D v13, le moteur de rendu Direct2D est utilisé par défaut sous Windows. Suivant votre configuration matérielle et logicielle, l'utilisation de ce moteur peut altérer le rendu de certains effets SVG tels que les ombres.
Dans ce cas, vous pouvez éventuellement désactiver Direct2D dans votre application à l'aide de la commande **FIXER PARAMETRE BASE**.

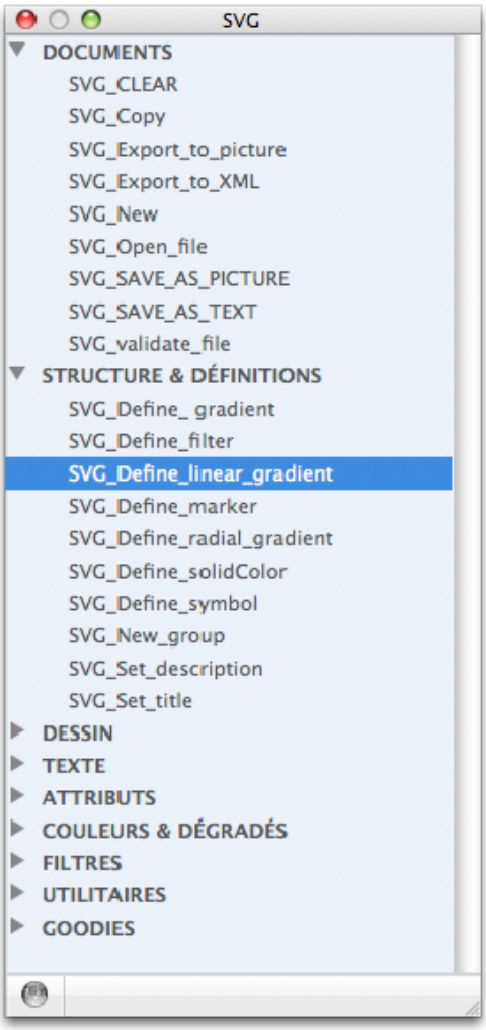
Outils de développement

Le composant 4D SVG propose un ensemble d'outils destinés à faciliter la saisie du code et la prévisualisation des graphiques SVG :

- la palette de syntaxe
- la palette de couleurs
- le visualisateur SVG

Palette de syntaxe

La palette de syntaxe liste les commandes du composant 4D SVG regroupées par thèmes :



La palette permet d'insérer les commandes du composant dans l'éditeur de méthodes par simple glisser-déposer. La commande est alors collée dans la méthode avec ses paramètres. Les paramètres optionnels sont préfixés d'un trait de soulignement.

Pour afficher la palette de syntaxe, vous pouvez soit :

- exécuter la méthode **SVGTool_Display_syntax**,
- cliquer sur le bouton **SVG** et choisir la commande **Syntaxe du composant SVG** dans la palette du composant 4D Pop si vous l'utilisez (cf. ci-dessous).

Palette de couleurs

La palette de couleurs affiche le nom et un échantillon de chaque couleur définie dans la norme SVG, ainsi qu'un curseur permettant de faire varier le taux d'opacité :

Vous pouvez utiliser cette palette pour insérer par glisser-déposer une référence de couleur SVG dans l'éditeur de méthodes de 4D. La couleur est insérée sous forme de chaîne incluant éventuellement le taux d'opacité (par exemple "lavender:30" pour la couleur lavande et une opacité de 30 %). Pour plus d'informations sur les références de couleurs, reportez-vous à la section **Couleurs SVG**.
Vous pouvez également glisser-déposer une couleur dans l'éditeur de formulaires de 4D. Cette action crée un carré de couleur sous forme d'image SVG statique.

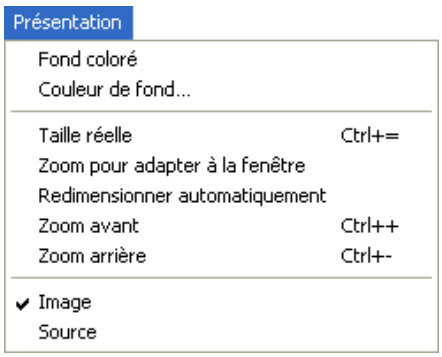
Pour afficher la palette de couleurs, il vous suffit d'exécuter la méthode **SVGTool_Display_colors**.

Visualisateur SVG

4D SVG propose un visualisateur SVG, particulièrement utile en phase de développement :

La fenêtre du visualisateur comporte deux pages, accessibles via les boutons **Image** et **Source** ou le menu Présentation :

- **Image** : cette page présente une zone de visualisation dans laquelle vous pouvez glisser-déposer ou ouvrir un fichier image SVG (via le menu Fichier). Vous pouvez également y afficher une référence SVG valide à l'aide de la commande *SVGTool_SHOW_IN_VIEWER*.
- **Source** : cette page permet de visualiser le code XML associé à l'image. Vous pouvez sélectionner et copier le code, mais vous ne pouvez pas le modifier.
  Lorsque la fenêtre est au premier plan, vous pouvez modifier plusieurs options d'affichage et enregistrer le fichier image sur disque via le menu Présentation :



Note : La page "Image" dispose d'un menu contextuel standard.

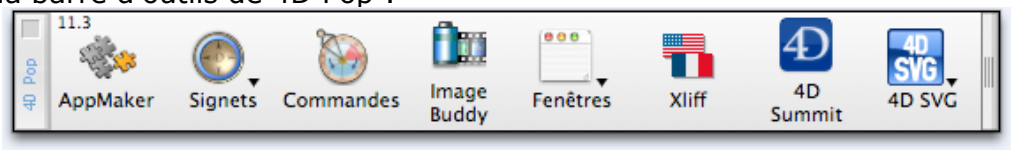
Pour afficher la fenêtre du visualisateur, vous pouvez soit :

- exécuter la méthode **SVGTool_Display_viewer**. Dans ce cas, la fenêtre s'affiche vide.
- appeler la méthode *SVGTool_SHOW_IN_VIEWER* en lui passant une référence SVG valide afin de prévisualiser l'image référencée (voir la description de la commande)
- cliquer sur le bouton **SVG** et choisir la commande **Visualisateur SVG** dans la palette du composant 4D Pop si vous l'utilisez (cf. ci-dessous).

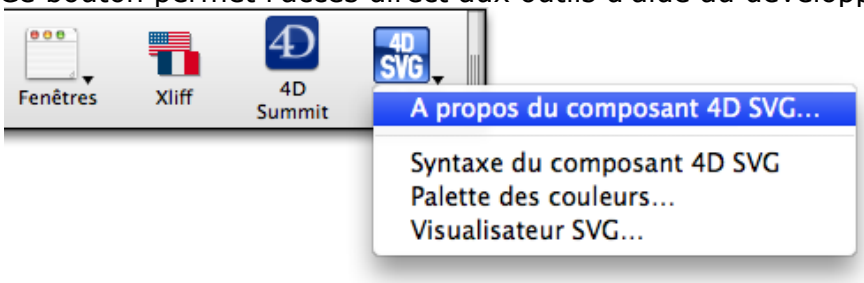
Intégration à 4D Pop

4D Pop est un ensemble de composants dédiés à la productivité du développeur et regroupés dans une barre d'outils qui s'intègre dans l'environnement de développement de 4D

Lorsque vous utilisez conjointement 4D Pop et 4D SVG, un nouveau bouton est disponible dans la barre d'outils de 4D Pop :



Ce bouton permet l'accès direct aux outils d'aide au développement de 4D SVG :



Note : 4DPop est inclus dans les outils additionnels sur l'installeur complet de 4D.

Précisions de syntaxe

Ref_SVG

La plupart des commandes du composant 4D SVG manipulent les structures SVG via des références de type **Ref_SVG**.

Une Ref_SVG est une expression de type Chaîne de 16 caractères. Elle identifie de façon unique une structure SVG chargée en mémoire. Il peut s'agir d'un document SVG chargé via les commandes *SVG_Copy*, *SVG_New*, *SVG_Open_picture* ou *SVG_Open_file*, ou de toute structure SVG manipulée par programmation (objet, filtre, tracé, etc.).

Une Ref_SVG est une référence XML. Toutes les références Ref_SVG peuvent être utilisées comme paramètres *refElément* des commandes XML DOM de 4D.

Une fois que vous n'en avez plus besoin, n'oubliez pas d'appeler la commande *SVG_CLEAR* avec les références Ref_SVG afin de libérer la mémoire.

Paramètres optionnels

Sauf mention contraire, les arguments numériques optionnels sont ignorés si leur valeur est égale à -1 et les arguments texte sont ignorés si la chaîne passée est vide.

Coordonnées

Sauf mention contraire, les paramètres de position (*x, y*) et de dimensions (*largeur*, *hauteur*, *rayon*) sont attendus dans le système de coordonnées utilisateur courant.

Attributs

- SVG_ADD_NAMESPACE
- SVG_GET_ATTRIBUTES
- SVG_Get_class
- SVG_Get_fill_brush
- SVG_Get_ID
- SVG_SET_ATTRIBUTES
- SVG_SET_ATTRIBUTES_BY_ARRAYS
- SVG_SET_CLASS
- SVG_SET_CLIP_PATH
- SVG_SET_DIMENSIONS
- SVG_SET_FILL_BRUSH
- SVG_SET_FILL_RULE
- SVG_SET_FILTER
- SVG_SET_ID
- SVG_SET_MARKER
- SVG_SET_OPACITY
- SVG_SET_ROUNDING_RECT
- SVG_SET_SHAPE_RENDERING
- SVG_SET_STROKE_BRUSH
- SVG_SET_STROKE_DASHARRAY
- SVG_SET_STROKE_LINECAP
- SVG_SET_STROKE_LINEJOIN
- SVG_SET_STROKE_MITERLIMIT
- SVG_SET_STROKE_WIDTH
- SVG_SET_TRANSFORM_FLIP
- SVG_SET_TRANSFORM_MATRIX
- SVG_SET_TRANSFORM_ROTATE
- SVG_SET_TRANSFORM_SCALE
- SVG_SET_TRANSFORM_SKEW
- SVG_SET_TRANSFORM_TRANSLATE
- SVG_SET_VIEWBOX
- SVG_SET_VIEWPORT_FILL
- SVG_SET_VISIBILITY
- SVG_SET_XY

SVG_ADD_NAMESPACE

SVG_ADD_NAMESPACE (objetSVG ; préfixe {; URI})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'objet SVG
préfixe	Texte	➡	Préfixe du namespace
URI	Texte	➡	URI du namespace

Description

La méthode **SVG_ADD_NAMESPACE** ajoute un attribut XML namespace à la racine de l'arbre DOM de la structure SVG référencée par *objetSVG*. Cette méthode vous permet notamment d'ajouter un namespace à un extrait de code SVG.

Passez dans le paramètre *préfixe* une chaîne contenant le préfixe de l'attribut namespace. Vous pouvez utiliser l'une des constantes suivantes :

- "svgNS" pour le namespace SVG standard (http://www.w3.org/2000/svg)
- "xlinkNS" pour le namespace standard XLink (http://www.w3.org/1999/xlink)

Dans ce cas, le paramètre *URI* est inutile.

Vous pouvez également passer un préfixe de namespace personnalisé dans le paramètre *préfixe* et son URI dans le paramètre *URI*. Dans ce cas, le paramètre *URI* est obligatoire, s'il est omis une erreur est générée.

Exemple

L'instruction suivante :

```
SVG_ADD_NAMESPACE($svgRef;"svgNS")
```

... ajoute le code suivant à la racine l'objet SVG :

```
<xmlns="http://www.w3.org/2000/svg">
```

SVG_GET_ATTRIBUTES

SVG_GET_ATTRIBUTES (objetSVG ; pointTabNoms ; pointTabValeurs)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence SVG
pointTabNoms	Pointeur	⇒	Tableau chaîne des libellés d'attributs
pointTabValeurs	Pointeur	⇒	Tableau chaîne des valeurs d'attributs

Description

La commande *SVG_GET_ATTRIBUTES* remplit les tableaux pointés par *pointTabNoms* et *pointTabValeurs* respectivement des noms et des valeurs des attributs de l'élément dont la référence est passée dans *objetSVG*. Si *objetSVG* n’est pas valide ou si cet attribut n'existe pas, une erreur est générée.

SVG_Get_class

SVG_Get_class (objetSVG {; nomsClasses}) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'objet SVG
nomsClasses	Pointeur	➡	Pointeur sur un tableau de noms de classes
Résultat	Texte	⮌	Nom(s) de classe(s)

Description

La commande **SVG_Get_class** retourne le nom de la ou des classe(s) pour une image SVG dont la référence est passée dans le paramètre *objetSVG*. Les noms des classes sont retournés sous forme de chaîne, chaque nom séparé par un espace.

Vous pouvez également récupérer les noms des classes sous forme de tableau : il suffit pour cela de passer un pointeur sur le tableau dans le paramètre optionnel *nomsClasses*.

Exemple

```
// définition de 2 styles
SVG_Define_style($Dom_SVG;".colored {fill: yellow; fill-opacity: 0.6; stroke: red; stroke-width: 8; stroke-opacity: 0.6}")
SVG_Define_style($Dom_SVG;".blue {fill: blue}")

// création d'un groupe et définition d'un style par défaut
$Dom_g:=SVG_New_group($Dom_SVG)
SVG_SET_CLASS($Dom_g;"colored blue")

TABLEAU TEXTE($tTxt_Classes;0)
$Txt_buffer:=SVG_Get_class($Dom_g;->$tTxt_classes)

// $Txt_buffer = "colored blue"
// $tTxt_classes{1} = "colored"
// $tTxt_classes{2} = "blue"
```

SVG_Get_fill_brush

SVG_Get_fill_brush (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence SVG
Résultat	Texte	⇐	Couleur de remplissage

Description

La commande **SVG_Get_fill_brush** retourne la couleur de remplissage de l'image SVG passée en référence dans le paramètre *objetSVG*. Si *objetSVG* ne possède pas d'attribut "fill" (couleur de remplissage), la commande retourne une chaîne vide.

SVG_Get_ID

SVG_Get_ID (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'élément SVG
Résultat	Chaîne	⇐	Nom de l'élément

Description

La commande*SVG_Get_ID* retourne la valeur de l'attribut 'id' de l'élément dont la référence est passée en paramètre.
Si *objetSVG* n’est pas valide ou si cet attribut n'existe pas, une erreur est générée.

SVG_SET_ATTRIBUTES

SVG_SET_ATTRIBUTES (objetSVG ; nomAttribut ; valeurAttribut {; nomAttribut2 ; valeurAttribut2 ; ... ; nomAttributN ; valeurAttributN})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
nomAttribut	Chaîne	➡	Nom de l'attribut à fixer
valeurAttribut	Chaîne	➡	Valeur de l'attribut

Description

La commande *SVG_SET_ATTRIBUTES* permet d'affecter un ou plusieurs attribut(s) personnalisé(s) à un objet SVG de référence *objetSVG*. Si le ou les attribut(s) existaient déjà, leurs valeurs sont remplacées par celles passées en paramètre.

Les attributs et leurs valeurs sont passés par couples de paramètres.

Exemple

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
SVG_SET_ATTRIBUTES($object;"style";"fill:red; stroke:blue; stroke-width:3")
```

SVG_SET_ATTRIBUTES_BY_ARRAYS

SVG_SET_ATTRIBUTES_BY_ARRAYS (objetSVG ; pointTabNoms ; pointTabValeurs)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
pointTabNoms	Pointeur	➡	Noms des attributs
pointTabValeurs	Pointeur	➡	Valeurs synchronisées des attributs

Description

La commande *SVG_SET_ATTRIBUTES_BY_ARRAYS* permet d'affecter un ou plusieurs attribut(s) personnalisé(s) à un objet SVG de référence *objetSVG*. Si le ou les attribut(s) existaient déjà, leurs valeurs sont remplacées par celles passées en paramètres.

Les attributs et leurs valeurs sont passés par l'intermédiaire de deux tableaux, sur lesquels pointent *pointTabNoms* et *pointTabValeurs*.

Exemple

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
TABLEAU TEXTE($attributes;0)
TABLEAU TEXTE($values;0)
AJOUTER A TABLEAU($attributes;"fill")
AJOUTER A TABLEAU($values;"red")
AJOUTER A TABLEAU($attributes;"stroke")
AJOUTER A TABLEAU($values;"blue")
AJOUTER A TABLEAU($attributes;"stroke-width")
AJOUTER A TABLEAU($values;"3")
SVG_SET_ATTRIBUTES_BY_ARRAYS($object;->$attributes;->$values)
```

SVG_SET_CLASS

SVG_SET_CLASS (objetSVG ; nomClasse)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'un élément SVG
nomClasse	Texte	⇒	Nom de la classe

Description

La commande *SVG_SET_CLASS* affecte la classe *nomClasse* à l'objet passé dans *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence valide.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/styling.html#StyleElement

Exemple

Reportez-vous à l'exemple de *SVG_Define_style*.

SVG_SET_CLIP_PATH

SVG_SET_CLIP_PATH (objetSVG ; idRognage)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
idRognage	Texte	➡	Nom du tracé de rognage

Description

La commande *SVG_SET_CLIP_PATH* assigne le tracé de rognage nommé *idRognage* à l'objet SVG désigné par *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence valide ou si le tracé le rognage n'est pas défini.

Référence :
http://www.yoyodesign.org/doc/w3c/svg1/masking.html#EstablishingANewClippingPath

Exemple 1

Définition d'un tracé de rognage circulaire ensuite attribué à une image :

```
//Définition d'un tracé circulaire
$Dom_clipPath:=SVG_Define_clipPath($Dom_SVG;"theClip")
$Dom_circle:=SVG_New_circle($Dom_clipPath;150;100;100)

//Création d'un groupe
$Dom_g:=SVG_New_group($Dom_SVG)

//Insertion d'une image
$Txt_path:=Dossier 4D(6)+"logo.svg"
LIRE FICHIER IMAGE($Txt_path;$Pic_buffer)
$Dom_picture:=SVG_New_embedded_image($Dom_g;$Pic_buffer)
SVG_SET_ID($Dom_picture;"MyPicture")

//Application du rognage au groupe
SVG_SET_CLIP_PATH($Dom_g;"theClip")
```

Exemple 2

La même image avec un tracé de rognage rectangulaire à coin arrondi :


```
 //Définition d'un tracé rectangulaire
$Dom_clipPath:=SVG_Define_clipPath($Dom_SVG;"theClip")
$Dom_rect:=SVG_New_rect($Dom_clipPath;5;10;320;240;10;10)

 //Création d'un groupe
$Dom_g:=SVG_New_group($Dom_SVG)

 //Insertion d'une image
$Txt_path:=Dossier 4D(6)+"logo.svg"
LIRE FICHIER IMAGE($Txt_path;$Pic_buffer)
$Dom_picture:=SVG_New_embedded_image($Dom_g;$Pic_buffer)
SVG_SET_ID($Dom_picture;"MyPicture")

 //Application du rognage au groupe
SVG_SET_CLIP_PATH($Dom_g;"theClip")
```

SVG_SET_DIMENSIONS

SVG_SET_DIMENSIONS (objetSVG ; largeur {; hauteur {; unité}})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
largeur	Entier long	➡	Dimension sur l'axe x
hauteur	Entier long	➡	Dimension sur l'axe y
unité	Chaîne	➡	Unité

Description

La commande *SVG_SET_DIMENSIONS* permet de fixer les dimensions de l'objet SVG de référence *objetSVG*.
Si ces attributs existaient déjà, leurs valeurs sont remplacées par celles passées en paramètres.

Si le paramètre *unité* est passé, il sera utilisé. Les valeurs attendues sont : px, pt, pc, cm, mm, in, em, ex ou %. Une valeur d'unité incorrecte génère une erreur. Si le paramètre est omis, les valeurs de *largeur* et *hauteur* sont attendues dans le système de coordonnées utilisateur.

Exemple

```
$svg :=SVG_New `Créer un nouveau document
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
SVG_SET_DIMENSIONS($object;-1;400) `Nouvelle hauteur
```

SVG_SET_FILL_BRUSH

SVG_SET_FILL_BRUSH (objetSVG ; couleur)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d’un élément SVG
couleur	Chaîne	➡	Expression couleur

Description

La commande *SVG_SET_FILL_BRUSH* permet de fixer la couleur de remplissage de l’objet SVG de référence *objetSVG*. Si cet attribut existait déjà, sa valeur est remplacée par celle passée en paramètre.

Pour plus d’informations sur les références de couleurs, reportez-vous à la section "**Couleurs SVG**".

Exemple

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
SVG_SET_FILL_BRUSH($object;"blue")
```

SVG_SET_FILL_RULE

SVG_SET_FILL_RULE (objetSVG ; modeRemplissage)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
modeRemplissage	Texte	➡	Mode de remplissage de l'objet

Description

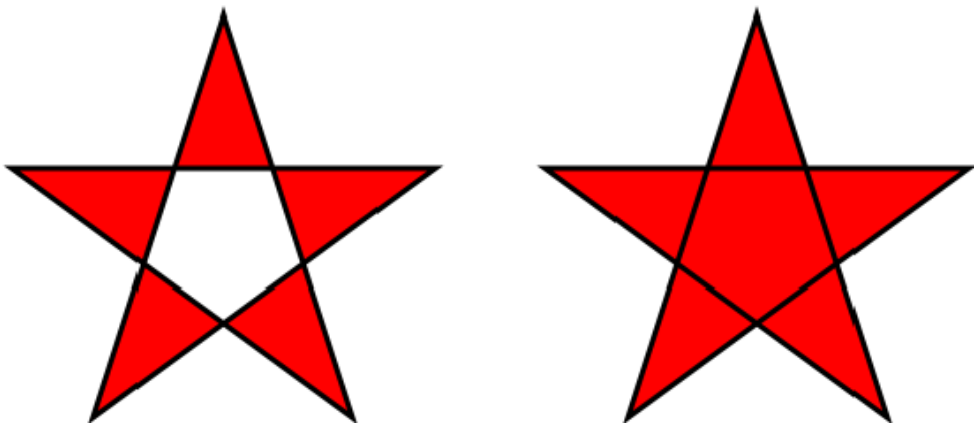
La commande *SVG_SET_FILL_RULE* permet de préciser le mode de remplissage de l'objet SVG désigné par *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence valide.

Le paramètre *modeRemplissage* doit contenir l'une des valeurs suivantes : "nonzero", "evenodd" ou "inherit". Dans le cas contraire, une erreur est générée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#FillRuleProperty

Exemple

Illustration des modes de remplissage pour obtenir ces tracés :



```
 //Création d'un tracé avec le mode de remplissage 'evenodd'
$Dom_path:=SVG_New_path($Dom_SVG;250;75)
SVG_PATH_LINE_TO($Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE($Dom_path)
SVG_SET_FILL_BRUSH($Dom_path;"red")
SVG_SET_STROKE_WIDTH($Dom_path;3)
SVG_SET_FILL_RULE($Dom_path;"evenodd")

 //Création d'un objet similaire avec le mode de remplissage 'nonzero'
$Dom_path:=SVG_New_path($Dom_SVG;250;75)
SVG_PATH_LINE_TO($Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE($Dom_path)
SVG_SET_FILL_BRUSH($Dom_path;"red")
SVG_SET_STROKE_WIDTH($Dom_path;3)
SVG_SET_FILL_RULE($Dom_path;"nonzero")
 //Déplacement horizontal
SVG_SET_TRANSFORM_TRANSLATE($Dom_path;300)
```

SVG_SET_FILTER

SVG_SET_FILTER (objetSVG ; id)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'élément SVG
id	Chaîne	⇒	Nom du filtre

Description

La commande *SVG_SET_FILTER* permet d'associer un filtre à l'objet de référence *objetSVG*. Si *objetSVG* n'est pas une référence valide, une erreur est générée. Si l'attribut existait déjà, sa valeur est remplacée.

Le paramètre *url* est le nom du filtre à utiliser, tel que défini avec la commande *SVG_Define_filter*. Si ce nom n'existe pas, une erreur est générée.

Exemple

Reportez-vous à l'exemple de la commande *SVG_Define_filter*.

SVG_SET_ID

SVG_SET_ID (objetSVG ; id)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'un élément SVG
id	Chaîne	⇒	Identifiant à attribuer à l'objet

Description

La commande *SVG_SET_ID* permet de fixer la propriété 'ID' de l'objet SVG de référence *objetSVG*. Si cet attribut existait déjà, sa valeur est remplacée par celle passée en paramètre.

L'ID d'un objet est utilisée pour référencer un objet. Cette référence peut ensuite être retrouvée grâce à la commande *SVG_Get_ID*. L'ID est également utilisée par la commande de 4D **SVG Chercher ID element par coordonnees** (voir la documentation de 4D).

Exemple

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";" white";2)
SVG_SET_ID($object;"bordure")
```

SVG_SET_MARKER

SVG_SET_MARKER (objetSVG ; id {; position})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'élément SVG
id	Chaîne	➡	Nom du marqueur
position	Chaîne	➡	Position du marqueur

Description

La commande **SVG_SET_MARKER** permet d'associer un marqueur à l'objet de référence *objetSVG* ou de supprimer un marqueur existant. Si *objetSVG* n'est pas la référence d'un élément 'line', 'path', 'polyline' ou 'polygon', une erreur est générée.
Si l'attribut existait déjà, sa valeur est remplacée.

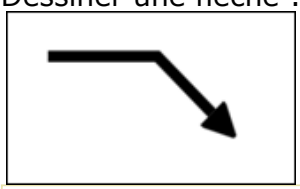
Le paramètre *id* est le nom d'un élément marqueur à utiliser tel que défini avec la commande **SVG_Define_marker**. Si ce nom n'existe pas, une erreur est générée.
Pour supprimer un marqueur existant, passez la chaîne "none" ou une chaîne vide dans le paramètre *id*.

Le paramètre optionnel *position* permet de fixer la position du marqueur par rapport à l'objet. Il est possible de placer un marqueur différent ou non au début, à la fin ou à tous les autres sommets d'un tracé. Les valeurs peuvent être :

- *start* pour placer un marqueur en début de tracé
- *end* pour placer un marqueur à la fin du tracé
- *middle* pour placer un marqueur à tous les sommets autres que le début et la fin.
- *all* pour placer un marqueur sur tous les sommets d'un tracé.
  Si ce paramètre est omis, le marqueur est placé à la fin du tracé.

Exemple 1

Dessiner une flèche :



```
$SVG:=SVG_New
 `Définir la flèche
$arrow:=SVG_Define_marker($SVG;"fleche";0;5;4;3;-1)
SVG_SET_VIEWBOX($arrow;0;0;10;10)
$path:=SVG_New_path($arrow;0;0)
SVG_SET_FILL_BRUSH($path;"black")
SVG_PATH_LINE_TO($path;10;5)
SVG_PATH_LINE_TO($path;0;10)
SVG_PATH_CLOSE($path)

$line:=SVG_New_path($SVG;100;75)
SVG_SET_STROKE_WIDTH($line;10)
SVG_PATH_LINE_TO($line;200;75)
SVG_PATH_LINE_TO($line;250;125)
```

```
`Mettre la flèche à la fin du tracé
SVG_SET_MARKER($line;"fleche")
```

Exemple 2

Dessiner un diagramme avec des marqueurs différents au début et à la fin :

```
$SVG:=SVG_New
SVG_SET_DEFAULT_BRUSHES("red";"red")

`Définir un cercle pour marquer les points
$point:=SVG_Define_marker($SVG;"pointMarker";2;2;3;3)
SVG_SET_VIEWBOX($point;0;0;4;4)
SVG_New_circle($point;2;2;1)

`Définir un carré pour le point de début
$start:=SVG_Define_marker($SVG;"startMarker";1;1;2;2)
SVG_New_rect($start;0;0;2;2)

Définir un triangle pour le point de fin
$end:=SVG_Define_marker($SVG;"endMarker";5;5;3;3;60)
SVG_SET_VIEWBOX($end;0;0;10;10)
SVG_New_regular_polygon($end;10;3)

TABLEAU ENTIER LONG($tX;0)
TABLEAU ENTIER LONG($tY;0)
`Axe des x
Boucle($Lon_i;0;200;20)
   AJOUTER A TABLEAU($tX;$Lon_i+10)
Fin de boucle
`Données
AJOUTER A TABLEAU($tY;100)
AJOUTER A TABLEAU($tY;100)
AJOUTER A TABLEAU($tY;30)
AJOUTER A TABLEAU($tY;30)
AJOUTER A TABLEAU($tY;80)
AJOUTER A TABLEAU($tY;60)
AJOUTER A TABLEAU($tY;10)
AJOUTER A TABLEAU($tY;40)
AJOUTER A TABLEAU($tY;50)
AJOUTER A TABLEAU($tY;70)
$line:=SVG_New_polyline_by_arrays($SVG;->$tX;->$tY;"red";"none";5)
`Disposer les marqueurs :
SVG_SET_MARKER($line;"startMarker";"start")
SVG_SET_MARKER($line;"pointMarker";"middle")
SVG_SET_MARKER($line;"endMarker";"end")
```

SVG_SET_OPACITY

SVG_SET_OPACITY (objetSVG ; opacitéFond {; ligne})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
opacitéFond	Entier long	➡	Opacité (%)
ligne	Entier long	➡	Opacité (%)

Description

La commande *SVG_SET_OPACITY* permet de fixer l'opacité du remplissage et du trait de l'objet de référence *objetSVG*.
Si ces attributs existaient déjà, leurs valeurs sont remplacées par celles passées en paramètres.
Les valeurs attendues sont comprises entre 0 et 100.

Exemple

```
$svg :=SVG_New `Créer un nouveau document
$object:=SVG_New_rect($svg ;10;10;200;100;0;0;"red";"blue")
SVG_SET_OPACITY($object;-1;50) `Fixer l'opacité du trait à 50%
```

SVG_SET_ROUNDING_RECT

SVG_SET_ROUNDING_RECT (objetSVG ; arrondiX {; arrondiY})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d’un rectangle SVG
arrondiX	Entier long	➡	Rayon sur l'axe x
arrondiY	Entier long	➡	Rayon sur l'axe y

Description

La commande *SVG_SET_ROUNDING_RECT* permet de fixer les rayons de l'ellipse utilisée pour l'arrondi des coins d'un rectangle de référence *objetSVG*. Si ces attributs existaient déjà, leurs valeurs sont remplacées par celles passées en paramètres. Si *objetSVG* n’est pas la référence d’un rectangle, une erreur est générée.
Les valeurs sont attendues dans le système de coordonnées utilisateur.

Exemple

```
$svg :=SVG_New ` Créer un nouveau document
$object:=SVG_New_rect($svg ;10;10;200;100)
SVG_SET_ROUNDING_RECT($object;20) `Arrondir les angles
```

SVG_SET_SHAPE_RENDERING

SVG_SET_SHAPE_RENDERING (objetSVG ; typeRendu)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d’un élément SVG
typeRendu	Texte	➡	Type de rendu de l’élément

Description

La commande *SVG_SET_SHAPE_RENDERING* permet de fixer les compromis à faire pour le rendu des éléments graphiques de l’objet désigné par *objetSVG*. Si *objetSVG* n'est pas un objet SVG, une erreur est générée.

Le paramètre *typeRendu* doit contenir l’une des valeurs suivantes : "auto", "optimizeSpeed", "crispEdges", "geometricPrecision" ou "inherit". Dans le cas contraire, une erreur est générée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#ShapeRenderingProperty

SVG_SET_STROKE_BRUSH

SVG_SET_STROKE_BRUSH (objetSVG ; couleur)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d’un élément SVG
couleur	Chaîne	➡	Expression couleur

Description

La commande *SVG_SET_STROKE_BRUSH* permet de fixer la couleur utilisée pour les traits de l’objet SVG de référence *objetSVG*. Si cet attribut existait déjà sa valeur, est remplacée par celle passée en paramètre.

Pour plus d’informations sur les couleurs, reportez-vous à la section "**Couleurs SVG**".

Exemple

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
SVG_SET_STROKE_BRUSH($object;"red")
```

SVG_SET_STROKE_DASHARRAY

SVG_SET_STROKE_DASHARRAY (objetSVG ; tiret {; valeur}{; valeur2 ; ... ; valeurN})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
tiret	Réel	➡	Longueur du premier tiret
valeur	Entier long	➡	Longueur des blancs et des tirets

Description

La commande *SVG_SET_STROKE_DASHARRAY* permet de définir le motif de tirets et de blancs utilisé pour le liseré du tracé de l'objet SVG désigné par *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence SVG valide.

La valeur entière du paramètre *tiret* indique la longueur du premier tiret du motif pointillé. Si les paramètres *valeur* sont omis, le pointillé sera constitué d'une succession de tirets et de blancs de la même longueur.

La valeur décimale du paramètre *tiret*, si elle est non nulle, indique à partir de quelle distance débuter les tirets.

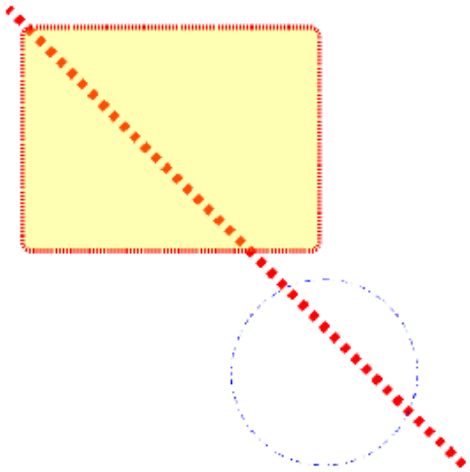
Si *tiret* vaut 0, le motif pointillé est supprimé.

Les paramètres *valeur* spécifient en alternance les longueurs des blancs et des tirets qui suivent le premier tiret. Si on fournit un nombre impair de valeurs (premier tiret compris), la liste des valeurs est répétée jusqu'à produire un nombre pair de valeurs.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#strokedasharrayProperty

Exemple

Tracés de pointillés :



```
 //Ligne
$Dom_line:=SVG_New_line($Dom_SVG;10;10;500;500)
SVG_SET_STROKE_WIDTH($Dom_line;10)
SVG_SET_STROKE_DASHARRAY($Dom_line;8,099)
SVG_SET_STROKE_BRUSH($Dom_line;"red")

 //Rectangle
$Dom_rect:=SVG_New_rect($Dom_SVG;25;30;320;240;10;10;"red";"yellow:30")
SVG_SET_STROKE_WIDTH($Dom_rect;5)
SVG_SET_STROKE_DASHARRAY($Dom_rect;2)
```

```
 //Cercle
$Dom_circle:=SVG_New_circle($Dom_SVG;350;400;100;"blue";"none")
SVG_SET_STROKE_DASHARRAY($Dom_circle;2;4;6;8)
```

SVG_SET_STROKE_LINECAP

SVG_SET_STROKE_LINECAP (objetSVG ; mode)

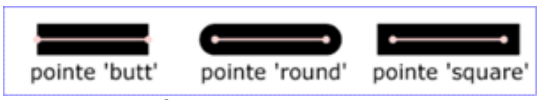
Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d’un élément SVG
mode	Chaîne	⇒	Mode de rendu

Description

La commande *SVG_SET_STROKE_LINECAP* permet de spécifier la forme des extrémités des tracés de l’objet SVG de référence *objetSVG*. Si cet attribut existait déjà, sa valeur est remplacée par celle passée en paramètre.

Le paramètre *mode* doit contenir l'une des chaînes suivantes, gérées par le SVG :

- *butt* (défaut) : standard
- *round* : arrondi
- *square* : carré
- *inherit* : hériter de l'objet parent



Si le paramètre *mode* contient une autre valeur, une erreur est générée.

SVG_SET_STROKE_LINEJOIN

SVG_SET_STROKE_LINEJOIN (objetSVG ; mode)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d’un élément SVG
mode	Chaîne	⇒	Mode de rendu

Description

La commande *SVG_SET_STROKE_LINEJOIN* permet de spécifier la forme des sommets des tracés de l’objet SVG de référence *objetSVG*. Si cet attribut existait déjà, sa valeur est remplacée par celle passée en paramètre.

Le paramètre *mode* doit contenir l'une des chaînes suivantes, gérées par le SVG :

- *miter* (défaut) : standard
- *round* : arrondi
- *bevel* : biseau
- *inherit* : hériter de l'objet parent

Si le paramètre *mode* contient une autre valeur, une erreur est générée.

SVG_SET_STROKE_MITERLIMIT

SVG_SET_STROKE_MITERLIMIT (objetSVG ; jointure)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇨	Référence d'un élément SVG
jointure	Entier long	⇨	Valeur de jointure

Description

La commande *SVG_SET_STROKE_MITERLIMIT* permet de définir l'étendue de la jointure entre le tracé et le liseré de l'objet SVG désigné par *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence valide.

Si le paramètre *jointure* est égal à -1, la valeur sera la valeur par défaut (4). Si le paramètre *jointure* est égal à 0, alors la définition de l'attribut est supprimée. Toute autre valeur < 0 provoquera une erreur.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#strokemiterlimitProperty

SVG_SET_STROKE_WIDTH

SVG_SET_STROKE_WIDTH (objetSVG ; tailleDuCrayon {; unité})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
tailleDuCrayon	Réel	➡	Epaisseur du tracé
unité	Chaîne	➡	Unité

Description

La commande *SVG_SET_STROKE_WIDTH* permet de fixer l'épaisseur des lignes de l'objet SVG de référence *objetSVG*. Si cet attribut existait déjà, sa valeur est remplacée par celle passée en paramètre.

Passez dans le paramètre *tailleDuCrayon* la valeur d'épaisseur des tracés. Le paramètre optionnel *unité* permet de définir l'unité utilisée. Vous pouvez passer l'une des valeurs suivantes : em, ex, px, pt, pc, cm, mm, in ou %. Si le paramètre *unité* est omis, la valeur *tailleDuCrayon* est attendue dans le système de coordonnées utilisateur.

Exemple

```
$svg :=SVG_New
SVG_SET_STROKE_WIDTH(SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2);10)
```

SVG_SET_TRANSFORM_FLIP

SVG_SET_TRANSFORM_FLIP (objetSVG ; horizontal {; vertical})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
horizontal	Booléen	➡	Miroir horizontal
vertical	Booléen	➡	Miroir vertical

Description

La commande *SVG_SET_TRANSFORM_FLIP* permet d'appliquer un miroir horizontal et/ou vertical à l'objet SVG de référence *objetSVG*.

Si le paramètre *horizontal* est égal à **Vrai**, un miroir horizontal est appliqué.
Si le paramètre *vertical* est égal à **Vrai**, un miroir vertical est appliqué

Exemple

Effet de miroir sur un objet texte :


```
svgRef:=SVG_New
SVG_SET_VIEWBOX(svgRef;0;0;400;200)
$tx:=SVG_New_text(svgRef;"4D";10;0;"";96)
SVG_SET_FONT_COLOR($tx;"blue") `Changer la couleur

 `Effet :
$tx:=SVG_New_text(svgRef;"4D";10;0;"";96) `Reprendre le même texte
SVG_SET_FONT_COLOR($tx;"lightblue") ` Changer la couleur
SVG_SET_TRANSFORM_FLIP($tx;Vrai) `Appliquer un miroir vertical
SVG_SET_TRANSFORM_SKEW($tx;-10) `Inclinaison
SVG_SET_TRANSFORM_TRANSLATE($tx;-17;-193) `Repositionner
```

SVG_SET_TRANSFORM_MATRIX

SVG_SET_TRANSFORM_MATRIX (objetSVG ; a ; b {; c ; d {; e ; f}})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
a	Entier long	➡	Elément a de la matrice de transformation
b	Entier long	➡	Elément b de la matrice de transformation
c	Entier long	➡	Elément c de la matrice de transformation
d	Entier long	➡	Elément d de la matrice de transformation
e	Entier long	➡	Elément e de la matrice de transformation
f	Entier long	➡	Elément f de la matrice de transformation

Description

La commande *SVG_SET_TRANSFORM_MATRIX* applique une transformation matricielle à l'objet SVG de référence *objetSVG*.

Ce type de transformation permet de combiner des transformations telles que, par exemple, une rotation et une translation.

Exemple


```
SVG_SET_TRANSFORM_MATRIX($ID;0,707;-0,707;0,707;0,707;255,03;111,21)
 `Equivaut à appliquer les 3 transformations suivantes :
SVG_SET_TRANSFORM_TRANSLATE($ID;50;90)
SVG_SET_TRANSFORM_ROTATE($ID;-45)
SVG_SET_TRANSFORM_TRANSLATE($ID;130;160)
```

SVG_SET_TRANSFORM_ROTATE

SVG_SET_TRANSFORM_ROTATE (objetSVG ; angle {; x ; y})

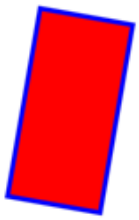
Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
angle	Entier long	➡	Angle de rotation
x	Entier long	➡	Coordonnée du centre de rotation sur l'axe x
y	Entier long	➡	Coordonnée du centre de rotation sur l'axe y

Description

La commande *SVG_SET_TRANSFORM_ROTATE* applique une rotation de la valeur *angle* à l'objet SVG de référence *objetSVG*.
L'angle de rotation est attendu en degrés, la rotation s'effectue dans le sens horaire.

Si les paramètres optionnels *x* et *y* ne sont pas passés, la rotation s'effectue par rapport à l'origine du système de coordonnées utilisateur courant. Si ces paramètres sont passés, la rotation s'effectue par rapport au point de coordonnées (*x*, *y*).

Exemple



```
svgRef:=SVG_New
  `Dessiner un rectangle rouge avec une bordure bleue
$rec:=SVG_New_rect($svg;150;50;200;400;0;0;"blue";"red";10)
  `Appliquer une rotation de 10° dans le sens horaire par rapport au centre
SVG_SET_TRANSFORM_ROTATE($rec;370;175;225)
```

SVG_SET_TRANSFORM_SCALE

SVG_SET_TRANSFORM_SCALE (objetSVG ; échelleX {; échelleY})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d’un élément SVG
échelleX	Réel	➡	Valeur sur l'axe x
échelleY	Réel	➡	Valeur sur l'axe y

Description

La commande *SVG_SET_TRANSFORM_SCALE* applique un changement d'échelle horizontale et/ou verticale à l’objet SVG de référence *objetSVG*.

Si la valeur *échelleX* est non nulle, l'objet est agrandi (valeur >1) ou diminué (0 < valeur < 1) horizontalement du nombre d'unités passé. La valeur 1 équivaut à ne pas changer l'échelle de l'objet.

Si le paramètre *échelleY* est passé, l'objet est agrandi (valeur >1) ou diminué (0 < valeur < 1) verticalement du nombre d'unités passé. La valeur 1 équivaut à ne pas changer l'échelle de l'objet. Si ce paramètre est omis, sa valeur est supposée égale à *échelleX*.

Exemple


```
$SVG:=SVG_New
$Text:=SVG_New_text($SVG;"Hello world!";5)
SVG_SET_TRANSFORM_SCALE($Text;3;12) `Zoom x*3 y*12
```

SVG_SET_TRANSFORM_SKEW

SVG_SET_TRANSFORM_SKEW (objetSVG ; horizontal {; vertical})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
horizontal	Entier long	➡	Valeur de l'inclinaison le long de l'axe x
vertical	Entier long	➡	Valeur de l'inclinaison le long de l'axe y

Description

La commande *SVG_SET_TRANSFORM_SKEW* spécifie une inclinaison horizontale et/ou verticale de l'objet SVG de référence *objetSVG*.

Si la valeur de *horizontal* est non nulle, l'objet sera incliné horizontalement du nombre d'unités passé, sinon elle est ignorée.
Si la valeur de *vertical* est non nulle, l'objet sera incliné verticalement du nombre d'unité passé.

Exemple


```
$svg :=SVG_New
 `Dessin du fond
SVG_New_rect($svg;0;0;270;160;10;10;"black";"gray")
 `Placer le texte...
$tx:=SVG_New_text($svg;"Hello world!";100;5;"";48)
 `en blanc
SVG_SET_FONT_COLOR($tx;"white")
 `Inclinaison
SVG_SET_TRANSFORM_SKEW($tx;-50;10)
```

SVG_SET_TRANSFORM_TRANSLATE

SVG_SET_TRANSFORM_TRANSLATE (objetSVG ; x {; y})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
x	Entier long	➡	Coordonnée sur l'axe x
y	Entier long	➡	Coordonnée sur l'axe y

Description

La commande *SVG_SET_TRANSFORM_TRANSLATE* spécifie une translation horizontale et/ou verticale de l'objet SVG de référence *objetSVG*.

Si la valeur *x* est non nulle, l'objet sera déplacé horizontalement du nombre d'unités passé, sinon elle est ignorée.
Si le paramètre *y* est fourni, l'objet sera déplacé verticalement du nombre d'unité passé.

Exemple

```
svgRef:=SVG_New
  `Dessiner un rectange rouge
$Object:=SVG_New_rect(svgRef;0;0;200;100;0;0;"black";"red")
  `Dessiner un carré en 0,0
$Object:=SVG_New_rect(svgRef;0;0;20;20)
  `Déplacer le carré en 150,50
SVG_SET_TRANSFORM_TRANSLATE($Object;150;50)
```

SVG_SET_VIEWBOX

SVG_SET_VIEWBOX (objetSVG ; x ; y ; largeur ; hauteur {; mode})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
x	Réel	➡	Position X du rectangle de visualisation
y	Réel	➡	Position Y du rectangle de visualisation
largeur	Réel	➡	Largeur du rectangle de visualisation
hauteur	Réel	➡	Hauteur du rectangle de visualisation
mode	Texte	➡	Adaptation au rectangle de visualisation

Description

La commande *SVG_SET_VIEWBOX* permet de définir le rectangle de visualisation de l'objet SVG de référence *objetSVG*. Si cet attribut existait déjà, sa valeur est remplacée par celle passée en paramètre.

Les valeurs sont attendues dans le système de coordonnées utilisateur.

Le paramètre optionnel *mode* permet d'indiquer si le graphique doit s'adapter, et comment, à la taille du rectangle de visualisation. La valeur attendue pour *mode* doit être une de celles reconnues par le SVG : 'none', 'xMinYMin', 'xMidYMin', 'xMaxYMin', 'xMinYMid', 'xMidYMid', 'xMaxYMid', 'xMinYMax', 'xMidYMax', 'xMaxYMax' ou 'true' (équivaut à xMidYMid).

Exemple

```
 `Créer un document SVG de 4x8cm
$svg:=SVG_New
SVG_SET_DIMENSIONS($SVG;4;8;"cm")
 `Déclarer le système de coordonnées utilisateur, ici 1 cm = 250 points utilisateur
SVG_SET_VIEWBOX($svg;0;0;1000;2000;"true")
```

SVG_SET_VIEWPORT_FILL

SVG_SET_VIEWPORT_FILL (objetSVG {; couleur {; opacité}})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'élémént SVG
couleur	Chaîne	➡	Couleur de remplissage
opacité	Entier long	➡	Pourcentage d'opacité

Description

La commande *SVG_SET_VIEWPORT_FILL* permet de fixer la couleur de fond d'un document SVG de référence *objetSVG*.

Si cet attribut existait déjà, sa valeur est remplacée par celle passée en paramètre. Si *objetSVG* est un élément SVG qui n'accepte pas cet attribut, une erreur est générée.

Le paramètre optionnel *couleur* indique la couleur à utiliser pour le fond de l'image. Si ce paramètre est omis ou contient une chaîne vide, c'est le blanc qui est utilisé. Pour plus d'informations sur les couleurs reportez-vous à la section **Couleurs SVG**.

Le paramètre optionnel *opacité* permet de préciser la valeur en pourcentage de l'opacité appliqué à ce remplissage. Si ce paramètre est omis et si aucune opacité n'était définie pour le document, la valeur 100% est utilisée.

SVG_SET_VISIBILITY

SVG_SET_VISIBILITY (objetSVG {; cacher})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
cacher	Booléen	➡	Vrai = montrer, Faux = cacher

Description

La commande *SVG_SET_VISIBILITY* cache ou montre l'objet SVG de référence *objetSVG*. Si *objetSVG* n'est pas la référence d'un objet pouvant être caché, une erreur est générée.

Si le paramètre optionnel *cacher* est égal à Vrai ou omis, l'objet est montré. S'il est égal à **Faux**, l'objet est caché.

Exemple

```
$svg :=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";" white";2)
SVG_SET_VISIBILITY($object;Faux) `L'objet est décrit mais ne sera pas rendu.
```

SVG_SET_XY

SVG_SET_XY (objetSVG ; x {; y})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
x	Entier long	➡	Coordonnée sur l'axe x
y	Entier long	➡	Coordonnée sur l'axe y

Description

La commande *SVG_SET_XY* permet de fixer les coordonnées du coin supérieur gauche de la région rectangulaire dans laquelle l'objet SVG de référence *objetSVG* se place.
Si ces attributs existaient déjà, leurs valeurs sont remplacées par celles passées en paramètres.
Si *objetSVG* est un élément SVG qui n'accepte pas cet attribut, une erreur est générée.
Les valeurs sont attendues dans le système de coordonnées utilisateur.

Exemple

```
$svg :=SVG_New `Créer un nouveau document
$object:=SVG_New_image($svg;"#Pictures/logo4D.png") `Placer le logo
SVG_SET_XY($object;10;40) `Modifier la position de l'image
```

Couleurs et dégradés

- Couleurs SVG
- SVG_Color_from_index
- SVG_Color_grey
- SVG_Color_RGB_from_CMYK
- SVG_Color_RGB_from_HLS
- SVG_Color_RGB_from_long
- SVG_FADE_TO_GREY_SCALE
- SVG_Filter_ColorMatrix
- SVG_GET_COLORS_ARRAY
- SVG_GET_DEFAULT_BRUSHES
- SVG_Get_named_color_value
- SVG_SET_BRIGHTNESS
- SVG_SET_DEFAULT_BRUSHES
- SVG_SET_HUE
- SVG_SET_SATURATION

Couleurs SVG

Définitions de couleurs

Le SVG reconnaît toutes les syntaxes alternatives pour les couleurs définies dans la norme CSS2. Les commandes du composant 4D SVG prennent en charge toutes ces syntaxes.
Une couleur peut être exprimée sous l'une des formes suivantes :

- Format RVB

Format	Exemple
#rgb	#f00
#rrggbb	#ff0000
rgb(r,g,b)	rgb(255,0,0)
	rgb(100%, 0%, 0%)

- Format mot-clé "couleur"
  Le SVG admet une liste étendue de mots-clés de noms de couleur, par exemple "red".

La liste des mots-clés ainsi que leur correspondance RVB figure dans les thèmes de **Constantes 4D SVG** : **SVG Colors (Names)** et **SVG Colors (RGB)**. Vous pouvez également visualiser cette liste et insérer directement des valeurs de couleurs via la Palette de couleurs de 4D SVG. Pour plus d'informations sur ce point, reportez-vous à la section **Outils de développement**.

None

Passez le mot-clé "**none**" dans le paramètre *coulPremierPlan* ou *coulArrièrePlan* si vous souhaitez ne pas définir de couleur de ligne ou de fond.
Le mot-clé "none" peut être utilisé avec la plupart des commandes SVG.

Opacité

Il est possible de spécifier l'opacité dans les expressions couleurs des commandes du composant en utilisant la syntaxe "couleur:opacité" où opacité est un nombre compris entre 0 (pas de couleur) et 100 (couleur complètement opaque). Ainsi "red:50" sera interprété comme un rouge à 50% d'opacité.

Dégradés

Les dégradés sont des transitions progressives de couleur le long d'un vecteur. Ces dégradés sont définis avec les commandes *SVG_Define_linear_gradient* et *SVG_Define_radial_gradient*.
Une fois définis, les dégradés sont utilisés par référence en utilisant la syntaxe "url(#NomDuDégradé)".
De même, il est possible de définir une couleur personnalisée associée à une opacité avec la commande *SVG_Define_solidColor*.

SVG_Color_from_index

SVG_Color_from_index (index) -> resultat

Paramètre	Type		Description
index	Entier long	⇒	Numéro de couleur
resultat	Text field	⇐	Couleur désignée par index

Description

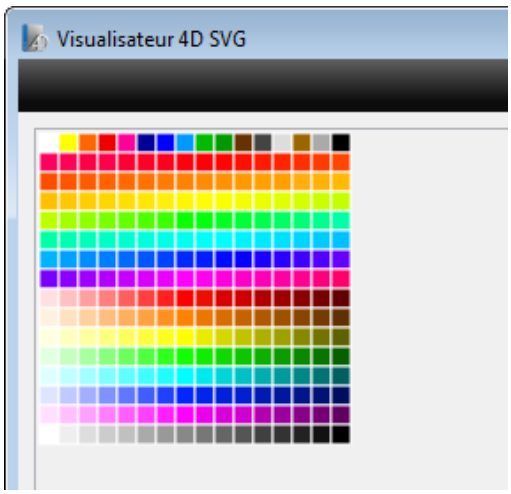
La méthode **SVG_Color_from_index** retourne la couleur SVG correspondant à la couleur 4D définie dans le paramètre *index*.

Le paramètre *index* désigne un numéro dans la palette de couleurs de 4D, numérotée de 1 à 256. Pour plus d'informations sur ce point, reportez-vous à la description de la commande de 4D **OBJET FIXER COULEUR**.

Exemple

Dans cet exemple, on recrée la palette de couleurs de 4D :

```
$Dom_svg:=SVG_New
$Lon_line:=0
Boucle($Lon_ii;0;15;1)
   $Lon_column:=0
   Boucle($Lon_i;1;16;1)
      $Txt_color:=SVG_Color_from_index(($Lon_ii*16)+$Lon_i)
      $Dom_rect:=SVG_New_rect($Dom_svg;$Lon_column;
      $Lon_line;11;11;0;0;"white";$Txt_color)
      $Lon_column:=$Lon_column+11
   Fin de boucle
   $Lon_line:=$Lon_line+11
Fin de boucle
SVGTool_SHOW_IN_VIEWER($Dom_svg)
```



SVG_Color_grey

SVG_Color_grey (pourcentage) -> Résultat

Paramètre	Type		Description
pourcentage	Entier	➡	Intensité du gris
Résultat	Chaîne	⮌	Chaîne couleur

Description

La commande *SVG_Color_grey* retourne une chaîne exprimant une couleur grise d'intensité *pourcentage*. La chaîne retournée est de la forme "rgb(rouge, vert, bleue)" où les 3 valeurs sont égales, syntaxe reconnue par les moteurs de rendu SVG.

Exemple

```
$txtColor:=SVG_Color_grey(60)
  `$txtColor vaut "rgb(102,102,102)"
```

SVG_Color_RGB_from_CMYK

SVG_Color_RGB_from_CMYK (cyan ; magenta ; jaune ; noir {; format}) -> Résultat

Paramètre	Type		Description
cyan	Entier long	➡	Valeur de cyan
magenta	Entier long	➡	Valeur de magenta
jaune	Entier long	➡	Valeur de jaune
noir	Entier long	➡	Valeur de noir
format	Entier long	➡	Format de la couleur
Résultat	Texte	⮌	Chaîne couleur

Description

La commande *SVG_Color_RGB_from_CMYK* retourne une chaîne exprimant la couleur correspondant aux paramètres quadrichromiques *cyan*, *magenta*, *jaune* et *noir* passés en arguments. La chaîne retournée est par défaut de la forme "RGB(rouge,vert,bleu)", syntaxe reconnue par les moteurs de rendu SVG.

cyan, *magenta*, *jaune* et *noir* sont des entiers longs compris entre 0 et 100%.

Le paramètre optionnel *format* permet de spécifier le format désiré pour la chaîne couleur retournée. Les valeurs sont :

Valeur	Format
1 (défaut)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

SVG_Color_RGB_from_HLS

SVG_Color_RGB_from_HLS (teinte ; luminosité ; saturation {; format}) -> Résultat

Paramètre	Type		Description
teinte	Entier long	➡	Valeur de la teinte
luminosité	Entier long	➡	Valeur de la luminosité
saturation	Entier long	➡	Valeur de la saturation
format	Entier long	➡	Format de la couleur
Résultat	Texte	⮌	Chaîne couleur

Description

La commande *SVG_Color_RGB_from_HLS* retourne une chaîne exprimant la couleur correspondant aux paramètres *teinte*, *luminosité* et *saturation* passés en arguments. La chaîne retournée est par défaut de la forme "RGB(rouge,vert,bleu)", syntaxe reconnue par les moteurs de rendu SVG.

teinte est un entier long compris entre 0 et 360°.

luminosité et *saturation* sont des entiers longs compris entre 0 et 100%.

Le paramètre optionnel *format* permet de spécifier le format désiré pour la chaîne couleur retournée. Les valeurs sont :

Valeur	Format
1 (défaut)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

SVG_Color_RGB_from_long

SVG_Color_RGB_from_long (couleur {; format}) -> Résultat

Paramètre	Type		Description
couleur	Entier long	➡	Valeur de la couleur
format	Entier	➡	Format de la couleur
Résultat	Chaîne	⮌	Chaîne couleur

Description

La commande *SVG_Color_RGB_from_long* retourne une chaîne exprimant la *couleur* passée en paramètre sous la forme "RGB(rouge, vert, bleue)", syntaxe reconnue par les moteurs de rendu SVG.

couleur est un entier long de 4 octets dont le format (0x00RRGGBB) est décrit ci-dessous (les octets sont numérotés de 0 à 3 de la droite vers la gauche) :

Octet	Description
2	Composante rouge de la couleur (0..255)
1	Composante verte de la couleur (0..255)
0	Composante bleue de la couleur (0..255)

Le paramètre optionnel *format* permet de spécifier le format désiré pour la chaîne couleur retournée. Les valeurs sont :

Valeur	Format
1 (défaut)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

Exemple

```
$txtColor:=SVG_Color_RGB_from_long($color)
  `$txtColor vaut "rgb(255,128,0)" si $color vaut 16744448 (orange)
```

SVG_FADE_TO_GREY_SCALE

SVG_FADE_TO_GREY_SCALE (objetSVG {; valeur})

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'objet SVG
valeur	Réel	⇒	Valeur de gris

Description

La commande **SVG_FADE_TO_GREY_SCALE** applique un filtre permettant de transformer l'échelle des gris de l'image SVG dont la référence est passée dans le paramètre *objetSVG*.

Vous pouvez passer une valeur d'échelle de gris à appliquer dans le paramètre optionnel *valeur*. Si vous ne passez pas ce paramètre, la transformation est effectuée selon la perception visuelle de la luminance (30% de rouge, 59% de vert et 11% de bleu).

SVG_Filter_ColorMatrix

SVG_Filter_ColorMatrix (objetSVG {; in ; result} {; type {; values}}) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence objet SVG
in	Texte	➡	Précise les entrées pour la primitive de filtre "ColorMatrix"
result	Texte	➡	Fournit une référence pour le résultat de sortie du filtre
type	Texte	➡	Indique le type d'opération de matrice
values	Texte	➡	Valeurs pour la matrice de transformation. Les valeurs numériques sont à passer avec le "." comme séparateur décimal.
Résultat	Ref_SVG	↩	Référence de l'objet SVG avec de nouvelles valeurs pour les couleurs

Description

La commande **SVG_Filter_ColorMatrix** applique une matrice de transformation sur chaque pixel de l'image source passée dans le paramètre *objetSVG* afin de produire un résultat avec de nouvelles couleurs.

Dans le paramètre *in*, vous pouvez passer une chaîne qui correspond à un précédent *result* ou un des six mots-clefs suivants :

- *SourceGraphic* : l'élément cible (image, forme, groupe etc.) auquel le filtre fait référence. Ce mot-clé représente les éléments graphiques manipulés par l'élément filtre.
- *SourceAlpha* : la couche sous la source *SourceGraphic*. Ce mot-clé représente les éléments graphiques manipulés par l'élément filtre.
- *BackgroundImage* : l'arrière-plan de la source *SourceGraphic*. Ce mot-clé représente un instantané de la couche sous la région manipulé par le filtre, au moment où l'élément filtre était invoqué.
- *BackgroundAlpha* : le canal alpha de la couche sous la source *SourceGraphic*. Identique à *BackgroundImage*, excepté que le canal alpha est utilisé.
- *FillPaint* : un pseudo-graphique égal à la taille de la région de filtre remplie par la propriété de remplissage de l'élément cible. Ce mot-clé représente la valeur de la propriété de remplissage sur l'élément cible pour l'effet de filtre.
- *StrokePaint* : un pseudo-graphique égal à la taille de la région du filtre remplie par la propriété de remplissage de l'élément cible. Ce mot-clé représente la valeur de la propriété 'Stroke' sur l'élément cible pour l'effet de filtre.

Si aucune valeur n'est passée et s'il s'agit de la première primitive de filtre, alors la source *SourceGraphic* est utilisée comme entrée. Si aucune valeur n'est passée et s'il s'agit d'une primitive de filtre subséquente, alors le filtre primitif utilisera le résultat du précédent filtre primitif en entrée.

Dans le paramètre *result, v*ous passez une référence pour le résultat en sortie du filtre référencé dans le paramètre *in* dans une utilisation subséquente de cette commande dans le même élément de filtre. Si aucune valeur n'est fournie, la sortie ne sera disponible qu'en réutilisation comme entrée implicite pour la primitive de filtre suivante, si cette dernière n'a aucune valeur pour son attribut *in.*

Dans le paramètre *type*, vous pouvez spécifier le type d'opération de matrice en passant un des attributs suivants :

- *matrix* : ce mot-clé signifie la fourniture d'une matrice complète de 5x4 (20) valeurs. Il fixe la couleur en utilisant la liste des valeurs passées dans le paramètre *values*. Permet de spécifier la valeur de chaque canal de sortie, à partir d'une combinaison du canal de sa couleur existante et du canal de la couche alpha.
- *saturate* : ajuste la saturation de tous les canaux de couleur RVB en utilisant un nombre réel de 0 à 1 passé dans le paramètre *values*.
- *hueRotate* : change la teinte de tous les canaux de couleur RVB par l'angle spécifié (en degré) dans le paramètre *values*,

- *luminanceToAlpha* : convertis les canaux rouge, vert et bleu vers la luminance indiquée. Les canaux RVB sont fixés en noir (0,0,0).

Si vous ne passez pas le paramètre *type*, par défaut c'est le type *Matrix* qui est appliqué.

Dans le paramètre *values*, vous passez les valeurs numériques en fonction du mot-clé passé dans le paramètre *type* :

- avec le mot-clé "*matrix*" : vous passez une matrice de 20 valeurs dans l'attribut *values*, séparées par un espace ou une virgule.
- avec le mot-clé "*saturate*" : l'attribut *values* admet une seule valeur de type réel (de 0 à 1). La valeur autorisée selon la spécification est 0-1, mais beaucoup de navigateurs acceptent des valeurs supérieures à >1 pour permettre la sur-saturation.
- avec le mot-clé "*hueRotate*" : l'attribut *values* admet une seule valeur de type réel, indiquant le degré de rotation.
- avec le mot-clé "*luminanceToAlpha*" : vous ne passez pas une valeur numérique. L'attribut *values* ne s'applique pas pour ce *type* qui n'utilise pas le canal alpha et le remplace par des valeurs égales à la luminance de l'image en entrée.

Si vous ne passez pas le paramètre *values*, le comportement par défaut dépend du mot-clé passé dans le paramètre *type* :

- avec le mot-clé "*matrix*" : par défaut les valeurs de la matrice identité sont utilisées.
- avec le mot-clé "*saturate*" : par défaut, la valeur est 1 (pas de changement).
- avec le mot-clé "*hueRotate*" : par défaut, la valeur est 0 (pas de changement).
- avec le mot-clé "*luminanceToAlpha*" : par défaut, ce paramètre n'est pas utilisé.

Note : Sous Windows, cette commande requiert la désactivation préalable de Direct2D (cf. constante Direct2D désactivé dans la description de la commande **FIXER PARAMETRE BASE**).

Exemple


```
C_TEXTE($Dom_filter;$Dom_node;$Dom_rect;$Dom_svg;$Txt_matrix)

SVG_SET_OPTIONS(SVG_Get_options ?+5)

$Dom_svg:=SVG_New

$Dom_filter:=SVG_Define_filter($Dom_svg;"Matrix")
$Txt_matrix:=\
".33 .33 .33 0 0 "\
+".33 .33 .33 0 0 "\
+".33 .33 .33 0 0 "\
+".33 .33 .33 0 0"


$Dom_node:=SVG_Filter_ColorMatrix($Dom_filter;"SourceGraphic";"";"matrix";$Txt_matrix)
```

```
$Dom_filter:=SVG_Define_filter($Dom_svg;"Saturate")
$Dom_node:=SVG_Filter_ColorMatrix($Dom_filter;"SourceGraphic";"";"saturate";"1.5")
   // another syntax for value
   //$Dom_node:=SVG_Filter_ColorMatrix
($Dom_filter;"SourceGraphic";"";"saturate";Chaine(1,5;"&xml"))

$Dom_filter:=SVG_Define_filter($Dom_svg;"HueRotate90")
$Dom_node:=SVG_Filter_ColorMatrix($Dom_filter;"SourceGraphic";"";"hueRotate";"90")

$Dom_filter:=SVG_Define_filter($Dom_svg;"LuminanceToAlpha")
$Dom_node:=SVG_Filter_ColorMatrix($Dom_filter;"SourceGraphic";"";"luminanceToAlpha")

$Dom_rect:=SVG_New_rect($Dom_svg;2;0;797;100;0;0;"none";"coral")

$Dom_rect:=SVG_New_rect($Dom_svg;2;100;797;100;0;0;"none";"coral")
SVG_SET_FILTER($Dom_rect;"Matrix")

$Dom_rect:=SVG_New_rect($Dom_svg;2;200;797;100;0;0;"none";"coral")
SVG_SET_FILTER($Dom_rect;"Saturate")

$Dom_rect:=SVG_New_rect($Dom_svg;2;300;797;100;0;0;"none";"coral")
SVG_SET_FILTER($Dom_rect;"HueRotate90")

$Dom_rect:=SVG_New_rect($Dom_svg;2;400;797;100;0;0;"none";"coral")
SVG_SET_FILTER($Dom_rect;"LuminanceToAlpha")

SVG_New_text($Dom_svg;"Pas de filtre";110;10;"Verdana";60;Gras;-1;"black")
SVG_New_text($Dom_svg;"Matrix";110;110;"Verdana";60;Gras;-1;"black")
SVG_New_text($Dom_svg;"Saturate";110;210;"Verdana";60;Gras;-1;"black")
SVG_New_text($Dom_svg;"HueRotate";110;310;"Verdana";60;Gras;-1;"black")
SVG_New_text($Dom_svg;"Luminance";110;410;"Verdana";60;Gras;-1;"black")

  //View the result
SVGTool_SHOW_IN_VIEWER($Dom_svg)

  //SVG_SAVE_AS_TEXT($Dom_svg;System folder(Desktop)+"export.svg")

  //Don't forget to clear the memory
SVG_CLEAR($Dom_svg)
```

SVG_GET_COLORS_ARRAY

SVG_GET_COLORS_ARRAY (ptrTabNomsCouls)

Paramètre	Type		Description
ptrTabNomsCouls	Pointeur	➡	Pointeur vers le tableau recevant les noms de couleurs

Description

La commande *SVG_GET_COLORS_ARRAY* remplit le tableau pointé par le paramètre *ptrTabNomsCouls* avec le nom des couleurs reconnues par le SVG.

Référence : http://www.w3.org/TR/SVG/types.html#ColorKeywords

SVG_GET_DEFAULT_BRUSHES

SVG_GET_DEFAULT_BRUSHES (ligne {; fond})

Paramètre	Type		Description
ligne	Pointeur	⇨	Variable chaîne
fond	Pointeur	⇨	Variable chaîne

Description

La commande *SVG_GET_DEFAULT_BRUSHES* renvoie dans la variable pointée par *ligne* la couleur par défaut courante pour le dessin des lignes.

Si le paramètre optionnel *fond* est passé, la variable pointée par ce paramètre recevra la couleur par défaut courante utilisée pour les fonds.

Si elles n'ont pas été modifiées, ces couleurs sont respectivement le noir et le blanc.

Exemple

Reportez-vous à l'exemple de la commande *SVG_SET_DEFAULT_BRUSHES*.

SVG_Get_named_color_value

SVG_Get_named_color_value (nomCouleur {; composanteRGB}) -> Résultat

Paramètre	Type		Description
nomCouleur	Texte	→	Nom de la couleur SVG
composanteRGB	Texte	→	"R", "G" ou "B" pour indiquer la composante de couleur
Résultat	Entier long	←	Valeur de couleur

Description

La commande **SVG_Get_named_color_value** retourne la valeur de la couleur SVG dont le nom est spécifié dans le paramètre *nomCouleur*.

Dans le paramètre optionnel *composanteRGB*, vous pouvez passer soit "R" (red), "G" (green) ou"B" (blue) pour indiquer la composante de couleur spécifique dont vous voulez récupérer la valeur. Si vous ne passez pas ce paramètre, la commande retourne la valeur de couleur complète.

SVG_SET_BRIGHTNESS

SVG_SET_BRIGHTNESS (objetSVG ; luminance {; luminance2 ; luminance3})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence SVG
luminance	Réel	➡	Valeurs entre 0 et 1 pour assombrir, > 1 pour éclaircir, appliquées globalement ou seulement sur la composante rouge de la couleur.
luminance2	Réel	➡	Luminance pour la composante verte de la couleur
luminance3	Réel	➡	Luminance pour la composante bleue de la couleur

Description

La commande **SVG_SET_BRIGHTNESS** fixe la luminance de l'image SVG ou du conteneur référencé dans le paramètre *svgObject*.

Dans le paramètre *luminance* ("brightness") vous passez soit une valeur entre 0 et 1 pour assombrir la luminance, soit une valeur supérieure à 1 pour l'éclaircir. Si vous passez un seul paramètre *luminance*, le facteur luminance est appliqué à l'objet dans sa totalité.

Mais si vous passez les deux paramètres (optionnels) *luminance2* et *luminance3*, dans ce cas chaque valeur est appliquée à chaque composante couleur : *luminance* est appliquée à la composante "R" (rouge), *luminance2* est appliquée à la composante "G" (vert) et *luminance3* est appliquée à la composante "B" (bleue).

SVG_SET_DEFAULT_BRUSHES

SVG_SET_DEFAULT_BRUSHES (ligne {; fond})

Paramètre	Type		Description
ligne	Chaîne	➡	Couleur de ligne
fond	Chaîne	➡	Couleur de fond

Description

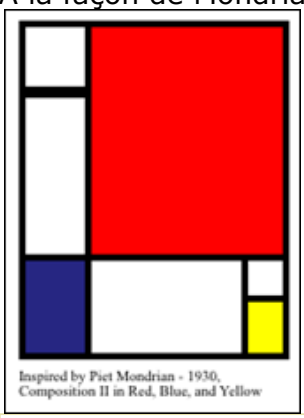
La commande *SVG_SET_DEFAULT_BRUSHES* permet de fixer les couleurs par défaut utilisées par le composant SVG.

Le paramètre *ligne* contient la nouvelle couleur qui sera utilisée pour les lignes. Le paramètre optionnel *fond* contient la nouvelle couleur à utiliser pour le dessin des fonds.

Passez une chaîne vide dans l'un ou l'autre de ces paramètres pour réinitialiser la valeur par défaut du composant, c'est-à-dire le noir et le blanc pour, respectivement, les lignes et le fond.

Exemple

A la façon de Mondrian...




```
$svg:=SVG_New
  `Fixer le couleurs par défaut
SVG_SET_DEFAULT_BRUSHES("black";"white")
  `Lignes de 4 points d'épaisseur
SVG_SET_STROKE_WIDTH($svg;4)
$g:=SVG_New_group($svg)
SVG_New_rect($g;2;2;40;40)
SVG_New_rect($g;2;45;40;100)
SVG_SET_FILL_BRUSH(SVG_New_rect($g;2;144;40;60);"midnightblue")
SVG_SET_FILL_BRUSH(SVG_New_rect($g;42;2;120;142);"red")
SVG_New_rect($g;42;144;95;60)
SVG_New_rect($g;137;144;25;25)
SVG_SET_FILL_BRUSH(SVG_New_rect($g;137;169;25;35);"yellow")
SVG_SET_TRANSFORM_TRANSLATE($g;10;10)
  `Légende
SVG_New_text($svg;"Inspired by Piet Mondrian - 1930,\rComposition II in Red, Blue, and
Yellow";10;220;"";9)
```

SVG_SET_HUE

SVG_SET_HUE (objetSVG ; hue)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'objet SVG
hue	Entier long	➡	Valeur de nuance

Description

La commande **SVG_SET_HUE** définit une valeur de nuance pour l'objet SVG désigné par le paramètre *objetSVG*. *objetSVG* doit être un conteneur SVG (svg, groupe, symbole, pattern, marqueur...) ou une image, sinon une erreur est générée.

Passez dans le paramètre *hue* une valeur située entre 0 et 360.

SVG_SET_SATURATION

SVG_SET_SATURATION (objetSVG ; saturation)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d’objet SVG
saturation	Entier long	⇒	Valeur de saturation

Description

La méthode **SVG_SET_SATURATION** définit une valeur de saturation pour l’objet SVG désigné par le paramètre *objetSVG*. *objetSVG* doit être un conteneur SVG (svg, groupe, symbole, pattern, marqueur...) ou une image, sinon une erreur est générée.

Passez dans le paramètre *saturation* une valeur située entre 0 et 100.

Dessin

- SVG_Add_object
- SVG_ADD_POINT
- SVG_New_arc
- SVG_New_circle
- SVG_New_ellipse
- SVG_New_ellipse_bounded
- SVG_New_embedded_image
- SVG_New_image
- SVG_New_line
- SVG_New_path
- SVG_New_polygon
- SVG_New_polygon_by_arrays
- SVG_New_polyline
- SVG_New_polyline_by_arrays
- SVG_New_rect
- SVG_New_regular_polygon
- SVG_PATH_ARC
- SVG_PATH_CLOSE
- SVG_PATH_CURVE
- SVG_PATH_LINE_TO
- SVG_PATH_MOVE_TO
- SVG_PATH_QCURVE
- SVG_Use

SVG_Add_object

SVG_Add_object (objetSVGCible ; objetSVGSource) -> Résultat

Paramètre	Type		Description
objetSVGCible	Ref_SVG	⇒	Référence de l'élément parent
objetSVGSource	Ref_SVG	⇒	Référence de l'objet à ajouter
Résultat	Ref_SVG	⇐	Référence de l'objet SVG

Description

La commande *SVG_Add_object* permet de placer dans le conteneur SVG désigné par *objetSVGCible* un objet SVG désigné par *objetSVGSource* et retourne sa référence. Le conteneur SVG peut être la racine du document ou toute autre référence à un objet SVG pouvant contenir ce type d'élément.

SVG_ADD_POINT

SVG_ADD_POINT (objetSVGParent ; x ; y {; x2 ; y2 ; ... ; xN ; yN})

Paramètre	Type		Description
objetSVGParent	Ref_SVG	⇒	Référence de tracé
x	Entier long	⇒	Coordonnée du nouveau point sur l'axe x
y	Entier long	⇒	Coordonnée du nouveau point sur l'axe y

Description

La commande *SVG_ADD_POINT* ajoute un ou plusieurs segments au tracé référencé par *objetSVGParent*. Le tracé peut être de type 'path', 'polyline' ou 'polygon'. Si *objetSVGParent* n'est pas n'est pas la référence d'un tracé de ce type, une erreur est générée.

Si plusieurs couples de coordonnées (*x*, *y*) sont passés, les différents points seront ajoutés successivement. Dans ce cas, si le dernier couple de coordonnées est incomplet (*y* manquant), il sera ignoré.

Exemples

Reportez-vous aux exemples de la commande **SVG_New_polyline**.

SVG_New_arc

SVG_New_arc (objetSVGParent ; x ; y ; rayon ; début ; fin {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
x	Entier long	→	Coordonnée du centre sur l'axe x
y	Entier long	→	Coordonnée du centre sur l'axe y
rayon	Entier long	→	Rayon du cercle
début	Entier long	→	Valeur en degré du début de l'arc
fin	Entier long	→	Valeur en degré de la fin de l'arc
coulPremierPlan	Chaîne	→	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	→	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	→	Epaisseur du tracé
Résultat	Ref_SVG	↩	Référence de l'arc

Description

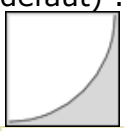
La commande *SVG_New_arc* crée un nouvel arc de cercle dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous aux commandes du thème **Couleurs et dégradés**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple 1

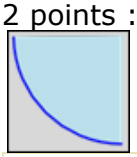
Dessin d'un arc de 0° à 90° (couleur de remplissage et de bordure, épaisseur du trait par défaut) :



```
svgRef:=SVG_New
objectRef:=SVG_New_arc(svgRef;100;100;90;90;180)
```

Exemple 2

Dessin d'un arc de cercle de 90° à 180° bleu clair avec un bord bleu et une épaisseur du trait de 2 points :



```
svgRef:=SVG_New
objectRef:=SVG_New_arc(svgRef;100;100;90;180;270;"blue";"lightblue";2)
```

SVG_New_circle

SVG_New_circle (objetSVGParent ; x ; y ; rayon {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
x	Entier long	→	Coordonnée du centre sur l'axe x
y	Entier long	→	Coordonnée du centre sur l'axe y
rayon	Entier long	→	Rayon du cercle
coulPremierPlan	Chaîne	→	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	→	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	→	Epaisseur du tracé
Résultat	Ref_SVG	↩	Référence du cercle

Description

La commande *SVG_New_circle* crée un nouveau cercle dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Le cercle est positionné et dimensionné d'après les coordonnées du centre (*x* et *y*) et le *rayon* passés en paramètres.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur du fond (pour plus d'informations sur les couleurs, reportez-vous aux commandes du thème **Couleurs et dégradés**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple 1

Dessin d'un cercle (couleur de remplissage et de bordure, épaisseur du trait par défaut) :

```
svgRef:=SVG_New
objectRef:=SVG_New_circle(svgRef;100;100;90)
```

Exemple 2

Dessin d'un cercle bleu clair avec un bord bleu et une épaisseur du trait de 2 points :

```
svgRef:=SVG_New
objectRef:=SVG_New_circle(svgRef;100;100;90;"blue";"lightblue";2)
```

SVG_New_ellipse

SVG_New_ellipse (objetSVGParent ; x ; y ; rayonX ; rayonY {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
x	Entier long	→	Coordonnée sur l'axe x du centre de l'ellipse
y	Entier long	→	Coordonnée du centre de l'ellipse sur l'axe y
rayonX	Entier long	→	Rayon sur l'axe x
rayonY	Entier long	→	Rayon sur l'axe y
coulPremierPlan	Chaîne	→	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	→	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	→	Epaisseur du tracé
Résultat	Ref_SVG	↩	Référence de l'ellipse

Description

La commande *SVG_New_ellipse* crée une nouvelle ellipse dans le conteneur SVG désigné par *objetSVGParent*. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

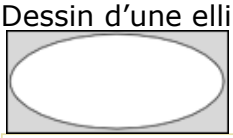
L'ellipse est positionnée et dimensionnée selon les valeurs de *x*, *y*, *largeur* et *hauteur*.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous aux commandes du thème **Couleurs et dégradés**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple 1

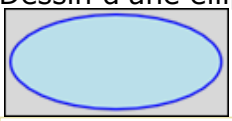
Dessin d'une ellipse (couleur de remplissage et de bordure, épaisseur du trait par défaut) :



```
svgRef:=SVG_New
objectRef:=SVG_New_ellipse(svgRef;100;50;90;40)
```

Exemple 2

Dessin d'une ellipse bleu clair avec un bord bleu et une épaisseur de trait de 2 points :



```
svgRef:=SVG_New
objectRef:=SVG_New_ellipse(svgRef;100;50;90;40;"blue";"lightblue";2)
```

SVG_New_ellipse_bounded

SVG_New_ellipse_bounded (objetSVGParent ; x ; y ; largeur ; hauteur {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
x	Entier long	→	Coordonnée du coin supérieur gauche sur l'axe x
y	Entier long	→	Coordonnée du coin supérieur gauche sur l'axe y
largeur	Entier long	→	Largeur du rectangle englobant
hauteur	Entier long	→	Hauteur du rectangle englobant
coulPremierPlan	Chaîne	→	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	→	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	→	Epaisseur du tracé
Résultat	Ref_SVG	↩	Référence de l'ellipse

Description

La commande *SVG_New_ellipse_bounded* crée une nouvelle ellipse dans le conteneur SVG désigné par *objetSVGParent*. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

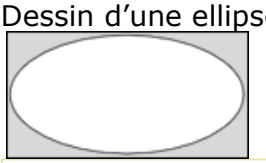
L'ellipse créée s'inscrit dans le rectangle défini par *x*, *y*, *largeur* et *hauteur*.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous aux commandes du thème **Couleurs et dégradés**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple 1

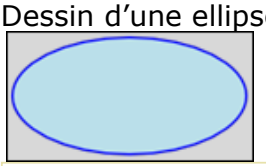
Dessin d'une ellipse (couleur de remplissage et de bordure, épaisseur du trait par défaut) :



```
svgRef:=SVG_New
objectRef:=SVG_New_ellipse_bounded(svgRef;10;10;200;100)
```

Exemple 2

Dessin d'une ellipse bleu clair avec un bord bleu et une épaisseur du trait de 2 points :



```
svgRef:=SVG_New
objectRef:=SVG_New_ellipse_bounded(svgRef;100;100;200;100;"blue";"lightblue";2)
```

SVG_New_embedded_image

SVG_New_embedded_image (objetSVGParent ; image {; x {; y}}{; codec}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
image	Image	➡	Image à inclure
x	Entier long	➡	Coordonnée du coin supérieur gauche sur l'axe x
y	Entier long	➡	Coordonnée du coin supérieur gauche sur l'axe y
codec	Texte	➡	Codec à utiliser
Résultat	Ref_SVG	↩	Référence de l'image

Description

La commande *SVG_New_embedded_image* permet d'inclure l'*image* dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

L'image sera encodée en base64 puis incluse dans le document.

Le paramètre *image* doit contenir une variable ou un champ image 4D.

Les paramètres optionnels *x* et *y* permettent de préciser la position du coin supérieur de l'image dans le conteneur SVG (valeur par défaut : 0).

Le paramètre optionnel *codec* permet de définir le codec à utiliser pour l'*image*. Par défaut, si ce paramètre est omis, le codec est ".png".

Exemple

Inclure l'image 'logo4D.png' située dans le dossier 'Resources' :

```
svgRef:=SVG_New
$Path:=Dossier 4D(Dossier Resources courant)+"logo4D.png"
LIRE FICHIER IMAGE($Path;$Picture)
Si(OK=1)
    objectRef:=SVG_New_embedded_image(svgRef;$Picture)
Fin de si
```

SVG_New_image

SVG_New_image (objetSVGParent ; url {; x ; y {; largeur ; hauteur}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
url	Chaîne	➡	Adresse de l'image
x	Entier long	➡	Coordonnée du coin supérieur gauche sur l'axe x
y	Entier long	➡	Coordonnée du coin supérieur gauche sur l'axe y
largeur	Entier long	➡	Largeur de l'image
hauteur	Entier long	➡	Hauteur de l'image
Résultat	Ref_SVG	↩	Référence de l'image

Description

La commande *SVG_New_image* permet de référencer une image située à l'adresse *url* dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Le paramètre *url* spécifie l'emplacement de l'image et peut prendre plusieurs formes :

- Un **URL local** (chemin d'accès de la forme "*file://...*"). Dans ce cas, l'image ne sera affichée que si le fichier est effectivement accessible au moment du rendu de l'objet. Cet URL local peut être relatif (de la forme "#Images/monImage.png"), dans ce cas la commande préfixera le chemin d'accès de celui du dossier **Resources** de la base hôte. Si les paramètres *largeur* et *hauteur* sont omis, ils seront calculés par la commande (dans ce cas l'exécution de la commande sera plus lente). Si le chemin relatif n'est pas valide, une erreur est générée.
- Un **URL non local** ("http://monSite.com/images/monImage.jpeg"). Dans ce cas, aucune vérification n'est effectuée sur la validité du lien et une erreur sera générée si les paramètres *largeur* et *hauteur* sont omis.
- Un **URL relatif** ("../picture.png"). Cette possibilité est particulièrement utile en client/serveur, lorsque les fichiers sont stockés dans le dossier "Resources". Les URLs relatifs peuvent débuter par :
  - "**/**", désignant le chemin "~/Resources/SVG/"
  - "**./**", désignant le chemin "~/Resources/"
  - "**../**", désignant le dossier de la base

Les paramètres optionnels *x* et *y* permettent de préciser la position du coin supérieur gauche de l'image dans le conteneur SVG (valeur par défaut : 0).

Les paramètres *largeur* et *hauteur* spécifient la taille du rectangle dans lequel sera affichée l'image et déterminent donc la taille et le ratio d'aspect de l'image. Ces paramètres ne sont optionnels que dans le cas d'une image référencée par un chemin relatif dans le dossier **Resources** de la base hôte. Si *largeur* et/ou *hauteur* vaut 0, l'image n'est pas rendue.

Exemple 1

Placer l'image 'logo4D.png' située dans le dossier 'Images' du dossier 'Resources' :

```
svgRef:=SVG_New
objectRef:=SVG_New_image(svgRef;"#Images/logo4D.png")
```

Exemple 2

Placer l’image ‘4dlogo.gif’ accessible dans le répertoire ‘images’ du site ‘4d.fr’ :

```
svgRef:=SVG_New
objectRef:=SVG_New_image(svgRef;"http://www.4d.fr/images/4dlogo.gif";20;20;39;53)
```

Exemple 3

Voici quelques exemples de référencement d'images à l'aide d'URLs relatifs :

```
SVG_New_image($Dom_svg;"./images/picture.png";10;10)
 // chemin relatif au dossier "Resources"
 // le code XML sera xlink:href="../images/picture.png"
```

```
SVG_New_image($Dom_svg;"../picture.png";70;180)
 // chemin relatif au dossier de la base
 // le code XML sera xlink:href="picture.png"
```

```
SVG_New_image($Dom_svg;"/sample pictures/picture.png";110;90;100;100)
 // chemin relatif au dossier "SVG" dans le dossier "Resources"
 // le code XML sera xlink:href="sample%20pictures/picture.gif"
```

SVG_New_line

SVG_New_line (objetSVGParent ; débutX ; débutY ; finX ; finY {; couleur {; tailleDuCrayon}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
débutX	Entier long	→	Position horizontale de début
débutY	Entier long	→	Position verticale de début
finX	Entier long	→	Position horizontale de fin
finY	Entier long	→	Position verticale de fin
couleur	Chaîne	→	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	→	Epaisseur du tracé
Résultat	Ref_SVG	←	Référence de ligne

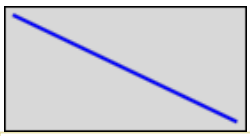
Description

La commande *SVG_New_line* crée une nouvelle ligne dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. L'objet est positionné d'après les coordonnées *débutX*, *débutY*, *finX* et *finY*. Le conteneur SVG peut être la racine du document ou toute autre référence à un objet SVG pouvant contenir ce type d'élément.

Le paramètre optionnel *couleur* contient le nom de la couleur de la ligne (pour plus d'informations sur les couleurs, reportez-vous à la section **Couleurs SVG**).
Le paramètre optionnel *tailleDuCrayon* définit la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple

Dessin d'une ligne bleue de trois pixels d'épaisseur :



```
svgRef:=SVG_New
objectRef:=SVG_New_line(svgRef;10;10;200;100;"blue";3)
```

SVG_New_path

SVG_New_path (objetSVGParent ; x ; y {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
x	Entier long	→	Coordonnée du début du tracé sur l'axe x
y	Entier long	→	Coordonnée du début du tracé sur l'axe y
coulPremierPlan	Chaîne	→	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	→	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	→	Epaisseur du tracé
Résultat	Ref_SVG	←	Référence du tracé

Description

La commande *SVG_New_path* débute un nouveau tracé dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Un tracé représente le contour d'une forme. Un tracé se décrit en faisant appel au concept de "point courant". Par analogie avec un dessin sur du papier, le point courant peut être assimilé à la position du crayon. Celle-ci peut changer et le contour d'une forme (ouverte ou fermée) peut être tracé en faisant glisser le crayon selon une ligne droite ou courbe.
Les tracés représentent la géométrie du contour d'un objet, définis selon les instructions des éléments *SVG_PATH_MOVE_TO* (établit un nouveau point courant), *SVG_PATH_LINE_TO* (dessine une droite), *SVG_PATH_CURVE* (dessine une courbe à l'aide d'une courbe de Bézier cubique), *SVG_PATH_ARC* (dessine un arc circulaire ou elliptique) et *SVG_PATH_CLOSE* (clôt la forme courante en dessinant une ligne jusqu'au dernier début de tracé). Il est possible d'avoir des tracés composés (c'est-à-dire un tracé avec plusieurs sous-tracés) qui permettent des effets comme un "trou de donut" dans des objets.

Les paramètres *x* et *y* permettent de préciser la position du début du tracé dans le conteneur SVG.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous à la section **Couleurs SVG**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple 1

Dessiner une ligne brisée fermée :

```
svgRef:=SVG_New
objectRef:=SVG_New_path(svgRef;20;20;"red";"none";5)
SVG_PATH_LINE_TO(objectRef;40)
SVG_PATH_LINE_TO(objectRef;40;40)
SVG_PATH_LINE_TO(objectRef;80;40;80;20;100;20;100;100;80;100;80;80;40;80;40;100;20;100)
SVG_PATH_CLOSE(objectRef)
```

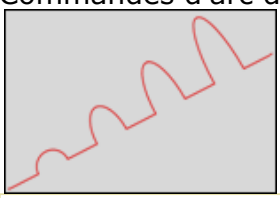
Exemple 2

Dessiner une courbe de Bézier :

```
svgRef:=SVG_New
objectRef:=SVG_New_path(svgRef;100;200;"aquamarine";"none";10)
SVG_PATH_CURVE(objectRef;250;200;100;100;250;100)
SVG_PATH_CURVE(objectRef;400;200;400;300)
```

Exemple 3

Commandes d'arc dans des données de tracé :



```
svgRef:=SVG_New
objectRef:=SVG_New_path(svgRef;20;300;"red";"none";2)
SVG_SET_OPTIONS(SVG_Get_options?-4)  `Passer en coordonnées relatives
SVG_PATH_LINE_TO(objectRef;50;-25)
Boucle($Lon_i;1;4;1)
   SVG_PATH_ARC(objectRef;25;25*$Lon_i;50;-25;-30)
   SVG_PATH_LINE_TO(objectRef;50;-25)
Fin de boucle
```

Exemple 4

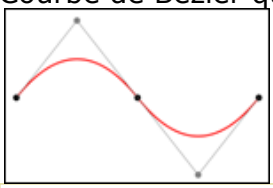
Tracé complexe (courbe de Bézier cubique) :

```
 `Création d'un nouvel élément SVG
$Txt_svg:=SVG_New(174,96;125,04;"Logo4D";"";Vrai)

 `Création d'un nouveau tracé
$Txt_path:=SVG_New_path($Txt_svg;150,665;13,021)
 `Définition des couleurs
SVG_SET_STROKE_BRUSH($Txt_path;"rgb(33,42,111)")
SVG_SET_FILL_BRUSH($Txt_path;"rgb(33,42,111)")
...
SVG_PATH_CURVE($Txt_path;-9,683;-6,54;-20,842;-8,888;-33,06;-10,462)
SVG_PATH_CURVE($Txt_path;-7,042;-0,915;-14,587;-0,877;-22,087;-0,877)
SVG_PATH_CURVE($Txt_path;-1,725;0;-4,312;-0,405;-5,761;0,24)
SVG_PATH_CURVE($Txt_path;-1,762;0;-5,092;-0,382;-6,479;0,24)
```

```
...
SVG_PATH_CURVE($Txt_path;181,489;70,216;177,236;30,976;150,665;13,021)
SVG_PATH_MOVE_TO($Txt_path;146,03;98,078)
...
SVG_PATH_CURVE($Txt_path;153,11;78,668;151,407;89,558;146,03;98,078)
```

Exemple 5

Courbe de Bézier quadratique :



```
 `Création d'un nouvel élément SVG
$svg:=SVG_New

 `Initialisation du trait noir et pas de remplissage
SVG_SET_DEFAULT_BRUSHES("";"none")

 `Dessin d'une courbe de Bézier quadratique en rouge
$qCurve:=SVG_New_path($svg;200;300)
SVG_SET_STROKE_BRUSH($qCurve;"red")
SVG_SET_STROKE_WIDTH($qCurve;5)
SVG_PATH_QCURVE($qCurve;400;50;600;300)
SVG_PATH_QCURVE($qCurve;1000;300)

 `Points finaux en noir
$g:=SVG_New_group($svg)
SVG_Set_description($g;"End points")
SVG_SET_DEFAULT_BRUSHES("black";"black")
SVG_New_circle($g;200;300;10)
SVG_New_circle($g;600;300;10)
SVG_New_circle($g;1000;300;10)

 `Points et lignes de contrôle en gris
$g:=SVG_New_group($svg)
SVG_Set_description($g;"Control points and lines from end points to control points")
SVG_SET_DEFAULT_BRUSHES(SVG_Color_grey(50);"none")
$path:=SVG_New_path($svg;200;300)
SVG_SET_STROKE_WIDTH($path;2)
SVG_PATH_LINE_TO($path;400;50;600;300;800;550;1000;300)
$gray:=SVG_Color_grey(50)  `grey 50%
SVG_SET_DEFAULT_BRUSHES($gray;$gray)
SVG_New_circle($g;400;50;10)
SVG_New_circle($g;800;550;10)
```

SVG_New_polygon

SVG_New_polygon (objetSVGParent {; points {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
points	Chaîne	➡	Tracé
coulPremierPlan	Chaîne	➡	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	➡	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	➡	Epaisseur du tracé
Résultat	Ref_SVG	↩	Référence du polygone

Description

La commande *SVG_New_polygon* crée une nouvelle forme fermée dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas une référence valide, une erreur est générée.

Le paramètre optionnel *points* permet de passer les points du tracé de la ligne tels qu'attendus par la norme SVG. Si ce paramètre est omis ou vide, les points pourront être définis avec la commande **SVG_ADD_POINTS**.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous aux commandes du thème **Couleurs et dégradés**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

SVG_New_polygon_by_arrays

SVG_New_polygon_by_arrays (objetSVGParent ; pointeurTabX ; pointeurTabY {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
pointeurTabX	Pointeur	➡	Coordonnées des points sur l'axe x
pointeurTabY	Pointeur	➡	Coordonnées des points sur l'axe y
coulPremierPlan	Chaîne	➡	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	➡	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	➡	Epaisseur du tracé
Résultat	Ref_SVG	↩	Référence du polygone

Description

La commande *SVG_New_polygon_by_arrays* dessine une forme fermée consistant en un jeu de segments de droites reliés dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Toutes les valeurs de coordonnées sont dans le système de coordonnées utilisateur.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous aux commandes du thème **Couleurs et dégradés**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple

Dessiner une étoile (couleur de bordure et épaisseur du trait par défaut) :

```
TABLEAU ENTIER LONG($tX;0)
TABLEAU ENTIER LONG($tY;0)

AJOUTER A TABLEAU($tX;129)
AJOUTER A TABLEAU($tY;10)
AJOUTER A TABLEAU($tX;158)
AJOUTER A TABLEAU($tY;96)
AJOUTER A TABLEAU($tX;248)
AJOUTER A TABLEAU($tY;96)
AJOUTER A TABLEAU($tX;176)
AJOUTER A TABLEAU($tY;150)
AJOUTER A TABLEAU($tX;202)
AJOUTER A TABLEAU($tY;236)
AJOUTER A TABLEAU($tX;129)
AJOUTER A TABLEAU($tY;185)
```

```
AJOUTER A TABLEAU($tX;56)
AJOUTER A TABLEAU($tY;236)
AJOUTER A TABLEAU($tX;82)
AJOUTER A TABLEAU($tY;150)
AJOUTER A TABLEAU($tX;10)
AJOUTER A TABLEAU($tY;96)
AJOUTER A TABLEAU($tX;100)
AJOUTER A TABLEAU($tY;96)

objectRef:=SVG_New_polygon_by_arrays(svgRef;->$tX;->$tY)
```

SVG_New_polyline

SVG_New_polyline (objetSVGParent {; points {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
points	Chaîne	➡	Tracé
coulPremierPlan	Chaîne	➡	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	➡	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	➡	Epaisseur du tracé
Résultat	Ref_SVG	↩	Référence de la ligne

Description

La commande *SVG_New_polyline* crée une nouvelle ligne brisée ouverte dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas une référence valide, une erreur est générée.

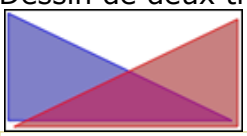
Le paramètre optionnel *points* permet de passer les points du tracé de la ligne tels qu'attendus par la norme SVG. Si ce paramètre est omis ou vide, les points pourront être définis avec la commande *SVG_ADD_POINT*.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous à la section **Couleurs SVG**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple

Dessin de deux triangles :



```
$polyline:=SVG_New_polyline($svg;"10,10 200,100 10,100 10,10";"blue";"blue:50")
$polyline:=SVG_New_polyline($svg;"";"red";"red:50")
SVG_ADD_POINT($polyline;205;15)
SVG_ADD_POINT($polyline;15;105)
SVG_ADD_POINT($polyline;205;105)
SVG_ADD_POINT($polyline;205;15)
```

SVG_New_polyline_by_arrays

SVG_New_polyline_by_arrays (objetSVGParent ; pointeurTabX ; pointeurTabY {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
pointeurTabX	Pointeur	➡	Coordonnées des points sur l'axe x
pointeurTabY	Pointeur	➡	Coordonnées des points sur l'axe y
coulPremierPlan	Chaîne	➡	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	➡	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	➡	Epaisseur du tracé
Résultat	Ref_SVG	↩	Référence de la ligne

Description

La commande *SVG_New_polyline_by_arrays* dessine une ligne brisée composée de segments de droite reliés entre eux dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Typiquement, les éléments 'polyline' définissent des formes ouvertes mais peuvent être utilisés pour des formes fermées. Dans ce cas, le dernier point défini doit être égal au premier.

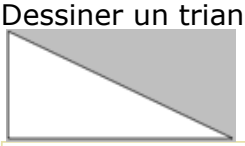
Toutes les valeurs de coordonnées sont situées dans le système de coordonnées utilisateur.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous à la section **Couleurs SVG**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple 1

Dessiner un triangle (couleur de bordure, épaisseur du trait par défaut) :



```
TABLEAU ENTIER LONG($tX;0)
TABLEAU ENTIER LONG($tY;0)

AJOUTER A TABLEAU($tX;10)
AJOUTER A TABLEAU($tY;10)
AJOUTER A TABLEAU($tX;200)
AJOUTER A TABLEAU($tY;100)
AJOUTER A TABLEAU($tX;10)
AJOUTER A TABLEAU($tY;100)
AJOUTER A TABLEAU($tX;10)
AJOUTER A TABLEAU($tY;10)

svgRef:=SVG_New
objectRef:=SVG_New_polyline_by_arrays(svgRef;->$tX;->$tY)
```

Exemple 2

Dessiner un diagramme en ligne :

```
TABLEAU ENTIER LONG($tX;0)
TABLEAU ENTIER LONG($tY;0)
  `Axe des X
Boucle($Lon_i;0;200;20)
    AJOUTER A TABLEAU($tX;$Lon_i)
Fin de boucle
  `Valeurs
AJOUTER A TABLEAU($tY;100)
AJOUTER A TABLEAU($tY;100)
AJOUTER A TABLEAU($tY;30)
AJOUTER A TABLEAU($tY;30)
AJOUTER A TABLEAU($tY;80)
AJOUTER A TABLEAU($tY;60)
AJOUTER A TABLEAU($tY;10)
AJOUTER A TABLEAU($tY;40)
AJOUTER A TABLEAU($tY;50)
AJOUTER A TABLEAU($tY;70)

objectRef:=SVG_New_polyline_by_arrays(svgRef;->$tX;->$tY;"crimson";"none";5)
```

SVG_New_rect

SVG_New_rect (objetSVGParent ; x ; y ; largeur ; hauteur {; arrondiX {; arrondiY {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
x	Entier long	→	X du coin supérieur gauche
y	Entier long	→	Y du coin supérieur gauche
largeur	Entier long	→	Largeur du rectangle
hauteur	Entier long	→	Hauteur du rectangle
arrondiX	Entier long	→	Arrondi horizontal
arrondiY	Entier long	→	Arrondi vertical
coulPremierPlan	Chaîne	→	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	→	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	→	Epaisseur du tracé
Résultat	Ref_SVG	↩	Référence du rectangle

Description

La commande *SVG_New_rect* crée un nouveau rectangle dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Le rectangle est positionné et dimensionné selon les valeurs de *x*, *y*, *largeur* et *hauteur*. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Les paramètres optionnel *arrondiX* et *arrondiY* permettent d'arrondir les angles suivant les valeurs indiqués. Si le paramètre *arrondiY* est omis (ou vaut -1), l'arrondi sera régulier. Passez -1 dans ces paramètres pour qu'ils soient ignorés par la commande.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous à la section **Couleurs SVG**).

Le paramètre optionnel *tailleDuCrayon* définit la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

Exemple 1

Dessin d'un rectangle (couleur de remplissage et de bordure, épaisseur du trait par défaut) :

```
svgRef:=SVG_New
objectRef:=SVG_New_rect(svgRef;10;10;200;100)
```

Exemple 2

Dessin d'un rectangle bleu muni d'une bordure rouge de trois pixels :

```
svgRef:=SVG_New
objectRef:=SVG_New_rect(svgRef;10;10;200;100;0;0;"red";"blue";3)
```

Exemple 3

Dessin d'un carré aux bords arrondis (couleur de remplissage et de bordure, épaisseur du trait par défaut) :

```
svgRef:=SVG_New
objectRef:=SVG_New_rect(svgRef;10;10;100;100;20)
```

Exemple 4

Dessin d'un rectangle bleu clair avec des bouts arrondis, un bord bleu (épaisseur du trait par défaut) :

```
svgRef:=SVG_New
objectRef:=SVG_New_rect(svgRef;10;10;200;100;-1;50;"blue";"lightblue")
```

SVG_New_regular_polygon

SVG_New_regular_polygon (objetSVGParent ; largeur ; nbCôtés {; x {; y {; coulPremierPlan {; coulArrièrePlan {; tailleDuCrayon}}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
largeur	Entier long	➡	Diamètre du cercle inscrit
nbCôtés	Entier long	➡	Nombre de côtés
x	Entier long	➡	Centre sur l'axe x du cercle circonscrit
y	Entier long	➡	Centre sur l'axe y du cercle circonscrit
coulPremierPlan	Chaîne	➡	Nom de la couleur ou du dégradé
coulArrièrePlan	Chaîne	➡	Nom de la couleur ou du dégradé
tailleDuCrayon	Réel	➡	Epaisseur du tracé
Résultat	Ref_SVG	⮌	Référence du polygone

Description

La commande *SVG_New_regular_polygon* dessine un polygone régulier ayant un nombre de côtés défini par *nbCôtés* inscrit dans le cercle de diamètre *largeur* dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Les paramètres optionnels *x* et *y* permettent de préciser le centre du cercle. S'ils sont omis, la figure sera dessinée dans le coin supérieur gauche du document.

Les paramètres optionnels *coulPremierPlan* et *coulArrièrePlan* contiennent respectivement le nom de la couleur de la ligne et de la couleur de fond (pour plus d'informations sur les couleurs, reportez-vous aux commandes du thème **Couleurs et dégradés**).

Le paramètre optionnel *tailleDuCrayon* contient la taille du crayon exprimée en pixels. Sa valeur par défaut est 1.

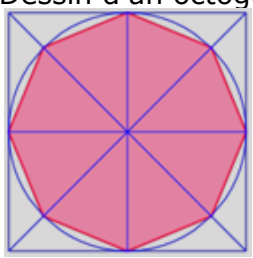
Exemple 1

Dessin d'un pentagone (couleur de remplissage et de bordure, épaisseur du trait par défaut) :

```
svgRef:=SVG_New
objectRef:=SVG_New_regular_polygon(svgRef;100;5)
```

Exemple 2

Dessin d'un octogone, de son cercle circonscrit et des lignes de tracé :



```
svgRef:=SVG_New
$width:=200
$sides:=8
objectRef:=SVG_New_regular_polygon(svgRef;$width;$sides;0;0;"crimson";"palevioletred";2)

$radius:=$width/2
objectRef:=SVG_New_rect(svgRef;0;0;$width;$width;0;0;"blue";"none")
objectRef:=SVG_New_line(svgRef;0;$radius;$width;$radius;"blue")
objectRef:=SVG_New_line(svgRef;$radius;0;$radius;$width;"blue")
objectRef:=SVG_New_line(svgRef;0;0;$width;$width;"blue")
objectRef:=SVG_New_line(svgRef;$width;0;0;$width;"blue")
objectRef:=SVG_New_circle(svgRef;$radius;$radius;$radius;"blue";"none")
```

SVG_PATH_ARC

SVG_PATH_ARC (objetSVGParent ; rayonX ; rayonY ; x ; y {; rotation {; cheminArc}})

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de tracé (élément path)
rayonX	Entier long	➡	Rayon de l'ellipse sur l'axe x
rayonY	Entier long	➡	Rayon de l'ellipse sur l'axe y
x	Entier long	➡	Coordonnée du point d'arrivée sur l'axe x
y	Entier long	➡	Coordonnée du point d'arrivée sur l'axe y
rotation	Entier long	➡	Valeur de rotation
cheminArc	Entier long	➡	Tracé de l'arc

Description

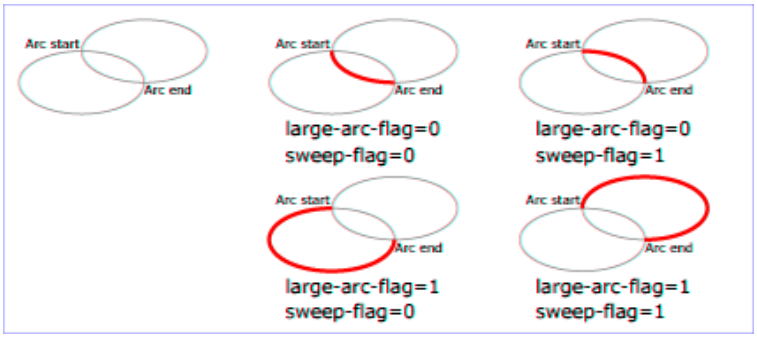
La commande *SVG_PATH_ARC* dessine un arc elliptique, du point courant jusqu'au point (*x*, *y*) à la fin du tracé référencé par *objetSVGParent*. Si *objetSVGParent* n'est pas la référence d'un tracé (élément 'path'), une erreur est générée.

La taille et l'orientation de l'ellipse sont définies par deux rayons (*rayonX*, *rayonY*) et une valeur de *rotation* sur l'axe x, qui indique la rotation de l'ellipse dans son ensemble par rapport au système de coordonnées courant.

Le paramètre optionnel *cheminArc* permet d'appliquer une combinaison de contraintes dans le but de définir la manière dont l'arc devra être dessiné : la contrainte large-arc-flag permet de choisir (ou non) le plus grand des deux arcs possibles (supérieur à 180°) et la contrainte sweep-flag définit la direction du tracé (angle positif ou négatif). Vous pouvez passer dans *cheminArc* l'une des valeurs suivantes, représentant les quatre combinaisons possibles des deux contraintes :

- 0 : large-arc-flag = 0, sweep-flag = 1
- 1 : large-arc-flag = 1, sweep-flag = 0
- 2 : large-arc-flag = 0, sweep-flag = 0
- 3 : large-arc-flag = 1, sweep-flag = 1

Lorsque large-arc-flag vaut 1, l'arc le plus grand est dessiné. Le plus petit est dessiné lorsque large-arc-flag vaut 0. Lorsque sweep-flag vaut 1, l'arc est dessiné avec un angle positif. L'arc est dessiné avec un angle négatif lorsque sweep-flag vaut 0. Le schéma suivant illustre les combinaisons possibles :



Par défaut, la valeur de *cheminArc* est 0 (large-arc-flag = 0 et sweep-flag = 1).

Exemple

Reportez-vous aux exemples de la commande *SVG_New_path*.

SVG_PATH_CLOSE

SVG_PATH_CLOSE (objetSVGParent)

Paramètre	Type		Description
objetSVGParent	Ref_SVG	⇨	Référence de tracé (élément path)

Description

La commande *SVG_PATH_CLOSE* ferme le sous-tracé courant référencé par *objetSVGParent* avec le dessin d'une ligne droite, du point courant jusqu'au point initial. Si *objetSVGParent* n'est pas la référence d'un tracé (élément 'path'), une erreur est générée.

Exemple

Reportez-vous aux exemples de la commande *SVG_New_path*.

SVG_PATH_CURVE

SVG_PATH_CURVE (objetSVGParent {; contrôleDébutX ; contrôleDébutY} ; contrôleFinX ; contrôleFinY ; x ; y)

Paramètre	Type		Description
objetSVGParent	Ref_SVG	⇒	Référence de tracé (élément path)
contrôleDébutX	Entier long	⇒	Coordonnée du point de contrôle sur l'axe x
contrôleDébutY	Entier long	⇒	Coordonnée du point de contrôle sur l'axe y
contrôleFinX	Entier long	⇒	Coordonnée du point de contrôle sur l'axe x
contrôleFinY	Entier long	⇒	Coordonnée du point de contrôle sur l'axe y
x	Entier long	⇒	Coordonnée du point d'arrivée sur l'axe x
y	Entier long	⇒	Coordonnée du point d'arrivée sur l'axe y

Description

La commande *SVG_PATH_CURVE* ajoute une courbe de Bézier cubique du point courant jusqu'au point de coordonnées (*x*, *y*) au tracé référencé par *objetSVGParent*. Si *objetSVGParent* n'est pas la référence d'un tracé (élément 'path'), une erreur est générée.

Les paramètres optionnels *contrôleDébutX* et *contrôleDébutY* permettent de spécifier la position du point de contrôle en début de courbe. S'ils sont omis, le premier point de contrôle est censé être le reflet du second point de contrôle de la commande précédente par rapport au point courant.

Les paramètres *contrôleFinX* et *contrôleFinY* permettent de spécifier la position du point de contrôle en fin de courbe.

Exemple

Reportez-vous aux exemples de la commande *SVG_New_path*.

SVG_PATH_LINE_TO

SVG_PATH_LINE_TO (objetSVGParent ; x {; y}{; x2 ; y2 ; ... ; xN ; yN})

Paramètre	Type		Description
objetSVGParent	Ref_SVG	⇒	Référence de tracé (élément path)
x	Entier long	⇒	Coordonnée sur l'axe x du nouveau point
y	Entier long	⇒	Coordonnée sur l'axe y du nouveau point

Description

La commande *SVG_PATH_LINE_TO* ajoute un ou plusieurs segments de droite au tracé référencé par *objetSVGParent*. Si *objetSVGParent* n'est pas la référence d'un tracé (élément 'path'), une erreur est générée.

Les paramètres *x* et *y* permettent de préciser la position du début du tracé dans le conteneur SVG.

- Si seul le paramètre *x* est fourni, la ligne sera tracée horizontalement depuis le point courant (xc, yc) jusqu'au point (x, yc).
- Si *x* et *y* sont passés, une ligne sera tracée depuis le point courant (xc, xc) jusqu'au point (x, y).
- Si plusieurs couples de coordonnées sont passés, les différents points seront ajoutés successivement. Dans ce cas, si le dernier couple de coordonnées est incomplet (*y* manquant), il sera ignoré.

Exemples

Reportez-vous aux exemples de la commande *SVG_New_path*.

SVG_PATH_MOVE_TO

SVG_PATH_MOVE_TO (objetSVGParent ; x ; y)

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de tracé (élément path)
x	Entier long	➡	Coordonnée sur l'axe x
y	Entier long	➡	Coordonnée sur l'axe y

Description

La commande *SVG_PATH_MOVE_TO* commence un nouveau sous-tracé au point de coordonnées (*x*, *y*) donné dans le tracé référencé par *objetSVGParent*. Si *objetSVGParent* n'est pas la référence d'un tracé (élément 'path'), une erreur est générée.

L'effet produit équivaut à soulever le "crayon" et le déplacer vers un nouvel emplacement. Le point courant devient le nouveau point de début qui sera pris en compte par la commande *SVG_PATH_CLOSE*.

Exemples

Reportez-vous aux exemples de la commande *SVG_New_path*.

SVG_PATH_QCURVE

SVG_PATH_QCURVE (objetSVGParent {; contrôleX ; contrôleY} ; x ; y)

Paramètre	Type		Description
objetSVGParent	Ref_SVG	⇒	Référence de tracé (élément path)
contrôleX	Entier long	⇒	Coordonnée du point de contrôle sur l'axe x
contrôleY	Entier long	⇒	Coordonnée du point de contrôle sur l'axe y
x	Entier long	⇒	Coordonnée du point d'arrivée sur l'axe x
y	Entier long	⇒	Coordonnée du point d'arrivée sur l'axe y

Description

La commande *SVG_PATH_QCURVE* ajoute une courbe de Bézier quadratique du point courant jusqu'au point de coordonnées (*x*, *y*) au tracé référencé par *objetSVGParent*. Si *objetSVGParent* n'est pas la référence d'un tracé (élément 'path'), une erreur est générée.

Les paramètres optionnels *contrôleX* et *contrôleY* permettent de spécifier la position du point de contrôle en début de courbe. S'ils sont omis, le premier point de contrôle est censé être le reflet du second point de contrôle de la commande précédente par rapport au point courant.

Exemple

Reportez-vous aux exemples de la commande *SVG_New_path*.

SVG_Use

SVG_Use (objetSVGParent ; id {; x ; y ; largeur ; hauteur {; mode}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
id	Chaîne	➡	Nom du symbole
x	Entier long	➡	Position X du rectangle de visualisation
y	Entier long	➡	Position Y du rectangle de visualisation
largeur	Entier long	➡	Largeur du rectangle de visualisation
hauteur	Entier long	➡	Hauteur du rectangle de visualisation
mode	Chaîne	➡	Adaptation au rectangle de visualisation
Résultat	Ref_SVG	↩	Référence de l'objet SVG

Description

La commande *SVG_Use* place une occurrence du symbole *id* dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG ou si *id* n'est pas le nom d'un objet du document SVG, une erreur est générée.

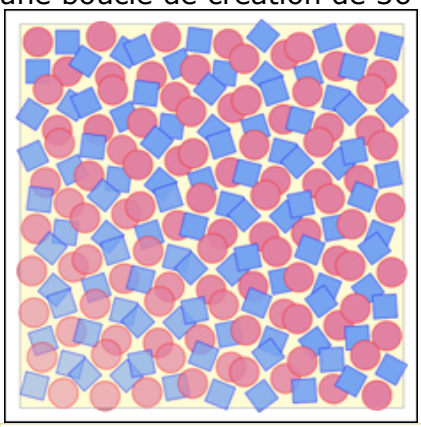
Un symbole est utilisé pour définir des objets graphiques, il n'est jamais rendu directement mais peut être instancié à l'aide de la commande *SVG_Use*.
Le paramètre *id* spécifie le nom du symbole.

Les paramètres optionnels *x*, *y*, *largeur* et *hauteur* spécifient le rectangle de la zone de visualisation ('viewBox').

Le paramètre optionnel *mode* permet d'indiquer si le graphique doit s'adapter, et comment, à la taille du rectangle de visualisation. (cf. commande *SVG_New*).

Exemple

Nous souhaitons définir un graphique composé de deux ronds rouges et deux carrés bleus. L'idée est d'utiliser ce graphique en faisant varier sa position, son opacité et sa rotation dans une boucle de création de 36 occurrences.



```
$SVG:=SVG_New
 `Dessin du fond
SVG_New_rect($SVG;20;20;650;650;0;0;"gray";"lemonchiffon")
 `Définition d'un symbole composé de 2 carrés et 2 cercles
$Symbol:=SVG_Define_symbol($SVG;"MySymbol";0;0;110;110;"true")
SVG_New_circle($Symbol;30;30;25;"red";"palevioletred")
SVG_New_rect($Symbol;10;60;40;40;0;0;"blue";"cornflowerblue")
SVG_New_rect($Symbol;60;10;40;40;0;0;"blue";"cornflowerblue")
SVG_New_circle($Symbol;80;80;25;"red";"palevioletred")
```

```
 `Dans un groupe...
$g:=SVG_New_group($SVG)
 `...positionné à 20 unités du coin supérieur gauche du document...
SVG_SET_TRANSFORM_TRANSLATE($g;20;20)
 `...placer 36 motifs en faisant varier la position, l'opacité et la rotation
Boucle($x;0;540;90) `6 colonnes
   Boucle($y;0;540;90) `6 lignes
      $use:=SVG_Use($g;"MySymbol";$x;$y;110;110)
      SVG_SET_OPACITY($use;100-($y/12)+($x/12)
      SVG_SET_TRANSFORM_ROTATE($use;($x*(18/50))+($y*(18/50));($x+55);($y+55))
   Fin de boucle
Fin de boucle
```

Documents

- SVG_CLEAR
- SVG_Copy
- SVG_Export_to_picture
- SVG_Export_to_XML
- SVG_New
- SVG_Open_file
- SVG_Open_picture
- SVG_SAVE_AS_PICTURE
- SVG_SAVE_AS_TEXT
- SVG_SET_DOCUMENT_VARIABLE
- SVG_Validate_file

SVG_CLEAR

SVG_CLEAR {(objetSVG)}

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'objet SVG

Description

La commande **SVG_CLEAR** libère l'espace mémoire occupé par l'objet SVG désigné par *objetSVG*.

objetSVG peut être un objet SVG racine (créé avec les commandes *SVG_New*, *SVG_Copy* ou *SVG_Open_file*) ou tout objet SVG valide.

Si le paramètre *objetSVG* n'est pas passé, la commande libère tous les arbres SVG créés par l'intermédiaire des commandes *SVG_New*, *SVG_Copy* ou *SVG_Open_file*. Cette syntaxe est utile durant la phase de développement, dans le cas où une référence SVG peut être créée mais l'espace mémoire non libéré, une erreur ayant empêché de terminer l'exécution de la méthode. Dans un développement final, toute référence SVG qui n'est plus utilisée doit être libérée avec la commande *SVG_CLEAR*.

SVG_Copy

SVG_Copy (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence de l'objet SVG à copier
Résultat	Ref_SVG	⮌	Référence du nouvel objet SVG

Description

La commande *SVG_Copy* crée un nouveau document SVG qui est une copie du document référencé par *objetSVG*.

La commande retourne une chaîne de 16 caractères (*Ref_SVG*) constituant la référence en mémoire de la structure virtuelle du nouveau document. Cette référence devra être utilisée avec les autres commandes du composant pour désigner le document.

Important : Une fois que vous n'en avez plus besoin, n'oubliez pas d'appeler la commande *SVG_CLEAR* avec cette référence afin de libérer la mémoire.

Exemple

```
svgRef:=SVG_New
...
svgRefCopy:=SVG_Copy(svgRef)
```

SVG_Export_to_picture

SVG_Export_to_picture (objetSVG {; typeExport}) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'objet SVG
typeExport	Entier long	➡	0=Ne pas stocker la source de données 1 (défaut)=Copier la source de données, 2=Prendre possession de la source de données
Résultat	Image	⮌	Image rendue par le moteur SVG

Description

La commande *SVG_Export_to_picture* commande retourne l'image décrite par la structure SVG référencée par *objetSVG*.

Le paramètre facultatif *typeExport* vous permet de définir la manière dont la source de données XML doit être prise en charge par la commande. Pour plus d'informations sur ce paramètre, reportez-vous à la description de la commande **SVG EXPORTER VERS IMAGE** de 4D. Si ce paramètre est omis, la valeur par défaut est 1, Copier source données XML.

Exemple

```
svgRef:=SVG_New(500;200;Test composant)
...
MyPicture:=SVG_Export_to_picture(svgRef;0)

SVG_CLEAR(svgRef)
```

SVG_Export_to_XML

SVG_Export_to_XML (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'objet SVG
Résultat	Texte	⮌	Texte XML du document SVG

Description

La commande *SVG_Export_to_XML* retourne le texte XML de description de la structure SVG référencée par *objetSVG*.

Exemple

```
svgRef:=SVG_New(500;200;Test composant)
...
MyText:=SVG_Export_to_XML(svgRef)
SVG_CLEAR(svgRef)
```

SVG_New

SVG_New {(largeur ; hauteur {; titre {; description {; rectangle {; affichage}}}})} -> Résultat

Paramètre	Type		Description
largeur	Entier long	➡	Largeur du document en points
hauteur	Entier long	➡	Hauteur du document en points
titre	Chaîne	➡	Titre du document
description	Chaîne	➡	Description
rectangle	Booléen	➡	Définir le rectangle de visualisation
affichage	Entier	➡	Format d'affichage de l'image
Résultat	Ref_SVG	↩	Référence de l'objet SVG

Description

La commande *SVG_New* crée un nouveau document SVG et retourne sa référence.

Les paramètres optionnels *largeur* et *hauteur* permettent de délimiter l'espace du document SVG. Ces deux paramètres doivent être exprimés en points utilisateur ('px'). Si vous souhaitez utiliser une autre unité, utilisez la commande *SVG_SET_DIMENSIONS*.

Les paramètres optionnels *titre* et *description* permettent d'associer au documents des informations de contenu.

Si vous passez **Vrai** dans le paramètre optionnel *rectangle*, le rectangle de visualisation (attribut 'viewBox') est automatiquement fixé à la taille du document créé.

Note : Il est possible de modifier les coordonnées du rectangle de visualisation du graphique et de régler plus précisément l'adaptation de l'image à ce dernier avec la commande *SVG_SET_VIEWBOX*.

Le paramètre optionnel *affichage* permet d'indiquer si le graphique doit s'adapter à la taille du document. Vous pouvez passer dans ce paramètre l'une des constantes du thème "**Formats d'affichage des images**" de 4D suivantes : Proportionnelle centrée ou Non tronquée.

La commande retourne une chaîne de 16 caractères (*Ref_SVG*) constituant la référence en mémoire de la structure virtuelle du document. Cette référence devra être utilisée avec les autres commandes du composant pour désigner le document.

Important : Une fois que vous n'en avez plus besoin, n'oubliez pas d'appeler la commande *SVG_CLEAR* avec cette référence afin de libérer la mémoire.

Exemple

```
svgRef:=SVG_New
svgRef:=SVG_New(500;200)
svgRef:=SVG_New(900;700;"Test composant";"Ceci est un exemple";Vrai;Non tronquée)
```

SVG_Open_file

SVG_Open_file (cheminAccès) -> Résultat

Paramètre	Type		Description
cheminAccès	Chaîne	⇒	Chemin d'accès du document SVG à ouvrir
Résultat	Ref_SVG	⇐	Référence du document ouvert

Description

La commande *SVG_Open_file* analyse (et valide avec la DTD) le document SVG se trouvant à l'emplacement désigné par le paramètre *chemin* et retourne une référence SVG (chaîne de 16 caractères) pour ce document.

Important : Une fois que vous n'en avez plus besoin, n'oubliez pas d'appeler la commande *SVG_CLEAR* avec cette référence afin de libérer la mémoire.

SVG_Open_picture

SVG_Open_picture (image) -> Résultat

Paramètre	Type		Description
image	Image	➡	Variable ou champ image 4D
Résultat	Ref_SVG	➡	Référence du document SVG

Description

La commande *SVG_Open_picture* analyse une image SVG et retourne une référence SVG pour cette image. Si *image* ne contient pas une image SVG, la commande retourne une chaîne vide.

Important : Une fois que vous n'en avez plus besoin, n'oubliez pas d'appeler la commande *SVG_CLEAR* avec cette référence afin de libérer la mémoire.

Exemple

```
LIRE FICHIER IMAGE("";$image)
Si(OK=1)
    $ref:=SVG_Open_picture($image)
    ...
    SVG_CLEAR($ref)
Fin de si
```

SVG_SAVE_AS_PICTURE

SVG_SAVE_AS_PICTURE (objetSVG ; nomFichier {; codec})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d’objet SVG
nomFichier	Chaîne	➡	Nom du document ou Chemin d’accès complet du document
codec	Chaîne	➡	Identifiant de codec d'image

Description

La commande **SVG_SAVE_AS_PICTURE** écrit le contenu de l’objet SVG désigné par *objetSVG* dans le fichier image désigné par *nomFichier*. Si *objetSVG* n’est pas un document SVG, une erreur est générée.

Vous pouvez passer dans *nomFichier* le chemin d'accès complet du fichier, ou uniquement son nom, auquel cas le fichier sera créé à côté du fichier de structure de la base. Si vous passez une chaîne vide ("") dans *nomFichier*, la boîte de dialogue standard d'enregistrement de fichiers apparaît, permettant à l'utilisateur de désigner le nom, l'emplacement et le format du fichier à créer.

Le paramètre optionnel *codec* vous permet de définir le format dans lequel l'image doit être sauvegardée. Si ce paramètre est omis, l’image est sauvegardée au format png.

La commande **SVG_SAVE_AS_PICTURE** met à jour la valeur de la variable éventuellement désignée par la commande **SVG_SET_DOCUMENT_VARIABLE**.

Exemple

```
svgRef:=SVG_New(500;200;Statistiques des ventes)
...
SVG_SAVE_AS_PICTURE(svgRef ;"test.png") //Sauvegarde
SVG_SAVE_AS_PICTURE(svgRef ;"test.gif";".gif")
SVG_CLEAR(svgRef )
```

SVG_SAVE_AS_TEXT

SVG_SAVE_AS_TEXT (objetSVG {; nomFichier})

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'objet SVG
nomFichier	Chaîne	⇒	Nom du document ou Chemin d'accès complet du document

Description

La commande **SVG_SAVE_AS_TEXT** écrit le contenu de l'objet SVG désigné par *objetSVG* dans le fichier disque désigné par *nomFichier*. Si *objetSVG* n'est pas un document SVG, une erreur est générée.

Vous pouvez passer dans *nomFichier* le chemin d'accès complet du fichier ou uniquement son nom, auquel cas le fichier sera créé à côté du fichier de structure de la base. Si vous passez une chaîne vide ("") dans *nomFichier* ou omettez ce paramètre, la boîte de dialogue standard d'enregistrement de fichiers apparaît, permettant à l'utilisateur de désigner le nom, l'emplacement et le format du fichier à créer.

La commande **SVG_SAVE_AS_TEXT** met à jour la valeur de la variable éventuellement désignée par la commande **SVG_SET_DOCUMENT_VARIABLE**.

Exemple

```
svgRef:=SVG_New(500;200;"Statistiques des ventes")
...
SVG_SAVE_AS_TEXT(svgRef;"test.svg")  //Le document est sauvegardé à côté de la structure
SVG_CLEAR(svgRef)
```

SVG_SET_DOCUMENT_VARIABLE

SVG_SET_DOCUMENT_VARIABLE (ptr)

Paramètre	Type		Description
ptr	Pointeur	➡	Pointeur vers la variable à définir

Description

La méthode **SVG_SET_DOCUMENT_VARIABLE** permet de définir un pointeur vers la variable de la base hôte qui sera mise à jour après chaque appel à **SVG_SAVE_AS_PICTURE** ou **SVG_SAVE_AS_TEXT**. Cette méthode doit être appelée une seule fois par session (par exemple dans une méthode d'initialisation).

Passez dans le paramètre *ptr* un pointeur vers la variable dont vous souhaitez suivre la valeur (généralement, la variable système **Document**).

Pour supprimer le lien, passez un pointeur Nil dans le paramètre *ptr*.

SVG_Validate_file

SVG_Validate_file (cheminAccès) -> Résultat

Paramètre	Type		Description
cheminAccès	Chaîne	⇒	Chemin d'accès du document SVG à valider
Résultat	Booléen	⇐	Vrai si le document correspond à la DTD

Description

La commande *SVG_Validate_file* tente de valider le document SVG stocké sur disque désigné par le paramètre *cheminAccès* avec la DTD (1.0). La commande retourne **Vrai** si le document est bien formé et **Faux** sinon.

Filtres

- Filtres SVG
- SVG_Filter_Blend
- SVG_Filter_Blur
- SVG_Filter_Offset

Filtres SVG

Les commandes du thème "**Filtres**" permettent de définir des effets de filtre applicables aux éléments SVG. Un effet de filtre consiste en une succession d'opérations graphiques, appliquées sur un graphique source, dont le produit est un graphique modifié.

Le résultat de l'effet de filtre est rendu sur l'appareil cible à la place du graphique source original.

Un filtre est défini grâce à la commande *SVG_Define_filter*, placée dans le thème "**Structure et Définitions**" et appliqué grâce à la commande *SVG_SET_FILTER*, placée dans le thème "**Attributs**". Les commandes du thème "**Filtres**" permettent de construire les opérations de filtrage ou "primitives de filtre".

SVG_Filter_Blend

SVG_Filter_Blend (refFiltre ; image ; imageFond {; mode {; nom}}) -> Résultat

Paramètre	Type		Description
refFiltre	Ref_SVG	➡	Référence de filtre
image	Chaîne	➡	Image source
imageFond	Chaîne	➡	Image de fond source
mode	Chaîne	➡	Mode de mélange
nom	Chaîne	➡	Cible de la primitive de filtre
Résultat	Ref_SVG	⬅	Référence de la primitive

Description

La commande *SVG_Filter_Blend* définit un filtre de composition pour le filtre *refFiltre* et retourne sa référence. Si *refFiltre* n'est pas une référence de filtre, une erreur est générée.

Ce filtre compose les deux sources, *imageFond* et *image*, à l'aide de modes de mélange couramment employés par les logiciels d'imagerie.

Le paramètre facultatif *mode* permet de définir le mode de combinaison des pixels utilisé pour le mélange (cf. spécification SVG). Sa valeur doit être : "normal" (valeur par défaut), "multiply", "screen", "darken" ou "lighten".

Le paramètre optionnel *nom* est le nom éventuellement assigné au résultat de cette primitive de filtre.

Note : A compter de 4D v14 R5, cette commande fonctionne sous Windows avec Direct2D activé en mode logiciel (cf. constante Direct2D Logiciel dans la description de la commande **FIXER PARAMETRE BASE**).

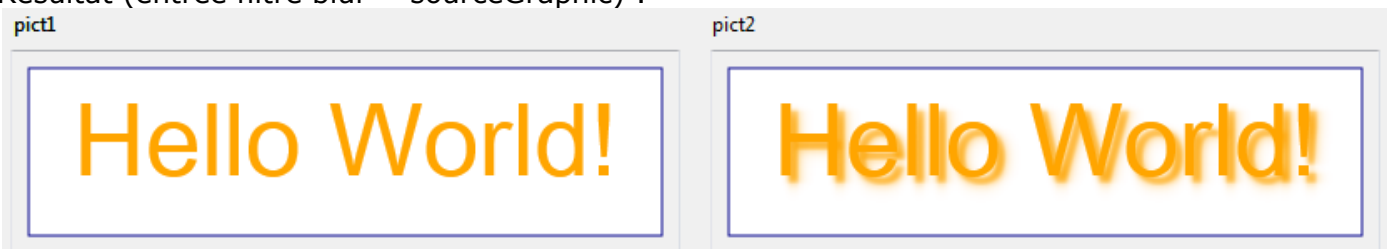
Exemple

Dans un formulaire, vous affichez deux images SVG identiques puis créez et affectez un filtre "blend" à l'image de droite. Ce filtre combine des filtres "offset" et "blur" :

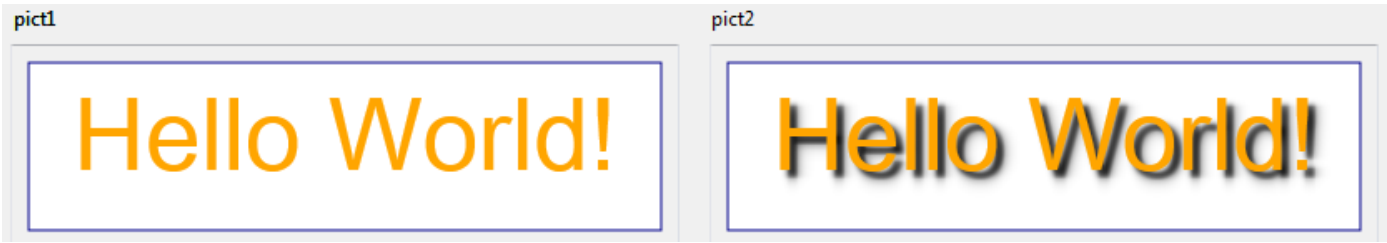
```
$root:=SVG_New(400;400;"filters test")  //définition de la première image (gauche)
$rect:=SVG_New_rect($root;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH($root;"orange")
$textAreaRef:=SVG_New_textArea($root;"Hello World!";10;10;380;100;"arial";60;Normal;Aligné au centre)
<>pict1:=SVG_Export_to_picture($root)  //affichage de la première image

$root2:=SVG_New(400;400;"filters test")  //définition de l'image de droite identique

  //création du filtre
$filter:=SVG_Define_filter($root2;"MyShadow")
$vGraph:=Vrai  //application sur la couche graphique - mettre à Faux pour la couche alpha
Si($vGraph)
   $ref1:=SVG_Filter_Blur($filter;2;"sourceGraphic";"blurResult")  //"blurResult" sera utilisé comme "entrée" du filtre offset
Sinon
   $ref1:=SVG_Filter_Blur($filter;2;"sourceAlpha";"blurResult")  //"blurResult" sera utilisé comme "entrée" du filtre offset
Fin de si
```

```
  //Ajout du filtre offset
$ref2:=SVG_Filter_Offset($filter;5;5;"blurResult";"alphaBlurOffset")
  //Ajout du filtre blend
$ref3:=SVG_Filter_Blend($filter;"sourceGraphic";"alphaBlurOffset";"normal";"finalFilter";)

$rect2:=SVG_New_rect($root2;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH($root2;"orange")
$textAreaRef2:=SVG_New_textArea($root2;"Hello World!";10;10;380;100;"arial";60;Normal;Aligné au centre)

SVG_SET_FILTER($textAreaRef2;"MyShadow")  //application du filtre final
<>pict2:=SVG_Export_to_picture($root2)  //affichage de la seconde image
```

Résultat (entrée filtre blur = sourceGraphic) :



Résultat (entrée filtre blur = sourceAlpha) :



SVG_Filter_Blur

SVG_Filter_Blur (refFiltre ; déviation {; entrée {; nom}}) -> Résultat

Paramètre	Type		Description
refFiltre	Ref_SVG	➡	Référence de filtre
déviation	Réel	➡	Déviation standard pour l'opération de flou
entrée	Chaîne	➡	Source de la primitive de filtre
nom	Chaîne	➡	Cible de la primitive de filtre
Résultat	Ref_SVG	↩	Référence de primitive

Description

La commande **SVG_Filter_Blur** définit un flou gaussien pour le filtre *refFiltre* et retourne sa référence. Si *refFiltre* n'est pas une référence de filtre, une erreur est générée.

Le paramètre *déviation* permet de définir la déviation standard pour l'opération de flou. Si le nombre est entier, la même déviation sera appliquée sur les axes X et Y. Si le nombre comporte une partie décimale, la partie entière représente la déviation à appliquer sur l'axe X et la partie décimale la déviation à appliquer sur l'axe Y.

Le paramètre optionnel *entrée* identifie la source graphique de la primitive de filtre. Vous pouvez passer :

- soit "sourceGraphic", désignant le graphique comme source du filtre (défaut),
- soit "sourceAlpha", désignant le canal alpha du graphique comme source du filtre.

Le paramètre optionnel *nom* est le nom éventuellement assigné au résultat de cette primitive de filtre.

Note : A compter de 4D v14 R5, cette commande fonctionne sous Windows avec Direct2D activé en mode logiciel (cf. constante Direct2D Logiciel dans la description de la commande **FIXER PARAMETRE BASE**).

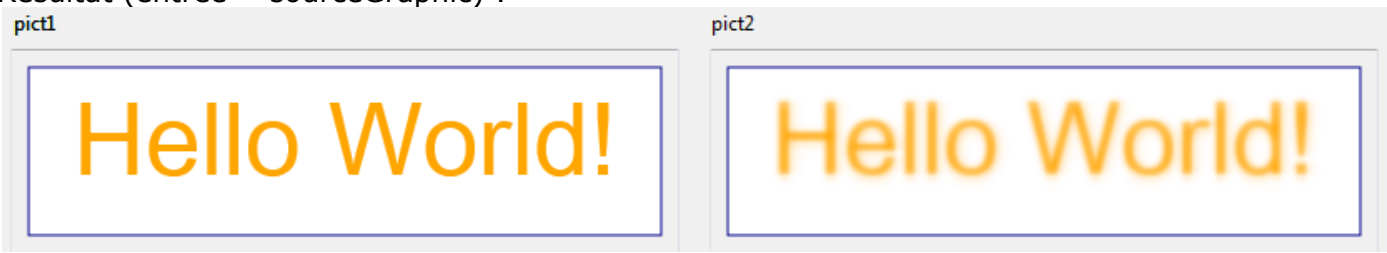
Exemple

Dans un formulaire, vous affichez deux images SVG identiques puis créez et affectez un filtre "blur" à l'image de droite :

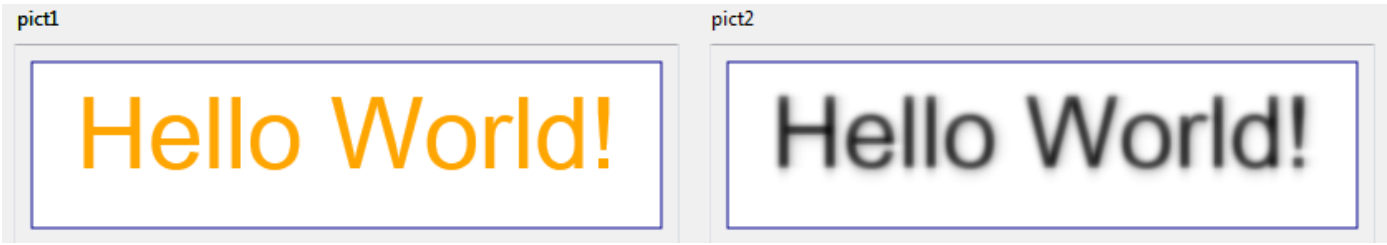
```
$root:=SVG_New(400;400;"filters test")  //définition de la première image (gauche)
$rect:=SVG_New_rect($root;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH($root;"orange")
$textAreaRef:=SVG_New_textArea($root;"Hello World!";10;10;380;100;"arial";60;Normal;Aligné au centre)
<>pict1:=SVG_Export_to_picture($root)  //affichage de la première image

$root2:=SVG_New(400;400;"filters test")  //définition de l'image de droite identique

  //création du filtre
$filter1:=SVG_Define_filter($root2;"blur")
  // définition du filtre
$vGraph:=Vrai  //application sur la couche graphique - mettre à Faux pour la couche alpha
Si($vGraph)
   SVG_Filter_Blur($filter1;Deviation{Deviation};"sourceGraphic")
Sinon
```

```
    SVG_Filter_Blur($filter1;Deviation{Deviation};"sourceAlpha")
  Fin de si

  $rect2:=SVG_New_rect($root2;10;10;380;100;0;0;"darkblue";"white";1)  //définition de l'image de droite identique
  SVG_SET_FILL_BRUSH($root2;"orange")
  $textAreaRef2:=SVG_New_textArea($root2;"Hello World!";10;10;380;100;"arial";60;Normal;Aligné au centre)

  SVG_SET_FILTER($textAreaRef2;"blur")  //application du filtre
  <>pict2:=SVG_Export_to_picture($root2)  //affichage de la seconde image
```

Résultat (entrée = sourceGraphic) :



Résultat (entrée = sourceAlpha) :



SVG_Filter_Offset

SVG_Filter_Offset (refFiltre ; dx {; dy {; entrée {; nom}}}) -> Résultat

Paramètre	Type		Description
refFiltre	Ref_SVG	➡	Référence de filtre
dx	Entier long	➡	Décalage sur l'axe x
dy	Entier long	➡	Décalage sur l'axe y
entrée	Chaîne	➡	Source de la primitive de filtre
nom	Chaîne	➡	Cible de la primitive de filtre
Résultat	Ref_SVG	↩	Référence de primitive

Description

La commande **SVG_Filter_Offset** définit un décalage pour le filtre *refFiltre* et retourne sa référence. Si *refFiltre* n'est pas une référence de filtre, une erreur est générée.

Le paramètre *dx* est la valeur du décalage horizontal.
Le paramètre optionnel *dy* est la valeur du décalage vertical.

Le paramètre optionnel *entrée* identifie la source graphique de la primitive de filtre. Vous pouvez passer :

- soit "sourceGraphic", désignant le graphique comme source du filtre (défaut),
- soit "sourceAlpha", désignant le canal alpha du graphique comme source du filtre.

Le paramètre optionnel *nom* est le nom éventuellement assigné au résultat de cette primitive de filtre.

Note : A compter de 4D v14 R5, cette commande fonctionne sous Windows avec Direct2D activé en mode logiciel (cf. constante Direct2D Logiciel dans la description de la commande **FIXER PARAMETRE BASE**).

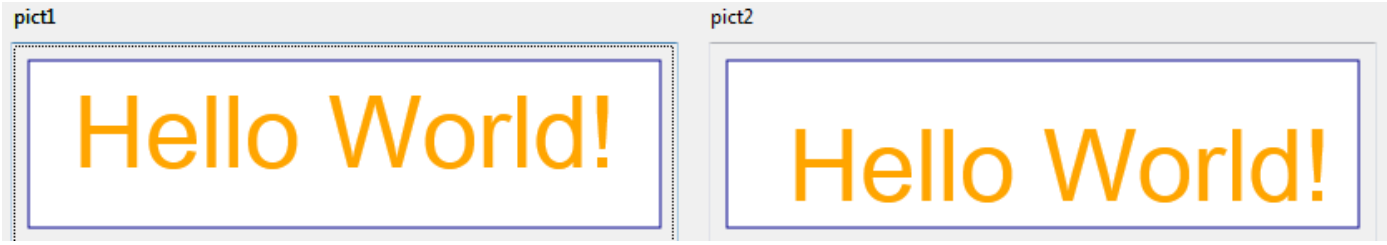
Exemple

Dans un formulaire, vous affichez deux images SVG identiques puis créez et affectez un filtre "offset" à l'image de droite :

```
$root:=SVG_New(400;400;"filters test")  //définition de la première image (gauche)
$rect:=SVG_New_rect($root;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH($root;"orange")
$textAreaRef:=SVG_New_textArea($root;"Hello World!";10;10;380;100;"arial";60;Normal;Aligné au centre)
<>pict1:=SVG_Export_to_picture($root)  //affichage de la première image

$root2:=SVG_New(400;400;"filters test")  //définition de l'image de droite identique
$rect2:=SVG_New_rect($root2;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH($root2;"orange")
$textAreaRef2:=SVG_New_textArea($root2;"Hello World!";10;10;380;100;"arial";60;Normal;Aligné au centre)

$filter:=SVG_Define_filter($root2;"Offset")  //création du filtre
SVG_Filter_Offset($filter;10;20)
SVG_SET_FILTER($textAreaRef2;"Offset")  //application du filtre
```

```
<>pict2:=SVG_Export_to_picture($root2) //affichage de la seconde image
```

Résultat :



Structure et Définitions

- SVG_Define_clip_path
- SVG_Define_filter
- SVG_Define_linear_gradient
- SVG_Define_marker
- SVG_Define_pattern
- SVG_Define_radial_gradient
- SVG_Define_shadow
- SVG_Define_solidColor
- SVG_Define_style
- SVG_DEFINE_STYLE_WITH_ARRAYS
- SVG_Define_symbol
- SVG_DELETE_OBJECT
- SVG_Get_default_encoding
- SVG_New_group
- SVG_SET_DEFAULT_ENCODING
- SVG_Set_description
- SVG_SET_PATTERN_CONTENT_UNITS
- SVG_SET_PATTERN_UNITS
- SVG_Set_title

SVG_Define_clip_path

SVG_Define_clip_path (objetSVGParent ; idRognage) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	⇒	Référence de l'élément parent
idRognage	Texte	⇒	Nom du tracé de rognage
Résultat	Ref_SVG	⇐	Référence du tracé de rognage

Description

La commande *SVG_Define_clip_path* définit un nouveau tracé de rognage dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas (ou n'appartient pas à) un document SVG, une erreur est générée.

Le paramètre *idRognage* désigne le nom du tracé de rognage. Le nom sera utilisé pour associer un tracé de rognage à un objet. Si un élément de même nom existe déjà dans le document, une erreur est générée.

Référence :
http://www.yoyodesign.org/doc/w3c/svg1/masking.html#EstablishingANewClippingPath

Exemple

Voir l'exemple de la commande *SVG_SET_CLIP_PATH*.

SVG_Define_filter

SVG_Define_filter (objetSVGParent ; id {; canevasX ; canevasY {; largeurCanevas ; hauteurCanevas {; unitéCanevas ; unitéFiltre}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
id	Chaîne	➡	Nom du symbole
canevasX	Entier long	➡	Coordonnée sur l'axe x
canevasY	Entier long	➡	Coordonnée sur l'axe y
largeurCanevas	Entier long	➡	Largeur du rectangle cible
hauteurCanevas	Entier long	➡	Hauteur du rectangle cible
unitéCanevas	Chaîne	➡	Système de coordonnées du canevas
unitéFiltre	Chaîne	➡	Système des valeurs du filtre
Résultat	Ref_SVG	↩	Référence du filtre

Description

La commande *SVG_Define_filter* définit un nouveau filtre dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Un filtre est une succession d'opérations graphiques qui seront appliquées sur l'élément cible. L'élément filtre n'est jamais rendu directement, il sera appliqué à un objet grâce à la commande *SVG_SET_FILTER*.

Le paramètre *id* spécifie le nom du filtre. Le nom sera utilisé pour associer un filtre à un objet. Si un élément de même nom existait, il est remplacé.

Les paramètres optionnels *canevasX*, *canevasY*, *largeurCanevas* et *hauteurCanevas* définissent une région rectangulaire du document sur laquelle ce filtre s'applique.

Le paramètre optionnel *unitéCanevas* définit le système de coordonnées pour les quatre paramètres précédents. Les valeurs attendues sont "*userSpaceOnUse*" ou "*objectBoundingBox*" (valeur par défaut).

Le paramètre optionnel *unitéFiltre* définit le système de coordonnées pour les longueurs et les propriétés de définition du filtre. Les valeurs attendues sont "*userSpaceOnUse*" (valeur par défaut) ou "*objectBoundingBox*".

Exemple

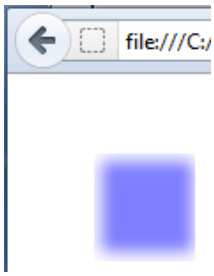
Dans cet exemple, nous souhaitons effectuer les opérations suivantes :

- création d'un rectangle fond bleu 50%
- création d'un filtre blur 4% et application au rectangle
- stockage du résultat dans un fichier svg sur disque.

```
$Dom_SVG:=SVG_New

 //création d'un rectangle fond bleu 50%
$Dom_rect:=SVG_New_rect($Dom_SVG;50;50;50;50;0;0;"blue:50";"blue:50")

 //création filtre blur (flou) 4%
$Dom_filter:=SVG_Define_filter($Dom_SVG;"blur")
SVG_Filter_Blur($Dom_filter;4)
SVG_Filter_Offset($Dom_filter;4)
```

```
 //application de ce filtre au rectangle
SVG_SET_FILTER($Dom_rect;"blur")

 //stockage du résultat dans un fichier svg
SVG_SAVE_AS_TEXT($Dom_SVG;Dossier systeme(Bureau)+"test.svg")

SVG_CLEAR($Dom_SVG)
```

Résultat :



SVG_Define_linear_gradient

SVG_Define_linear_gradient (objetSVGParent ; id ; couleurDébut ; couleurFin {; rotation {; spreadMethod {; x1 ; y1 ; x2 ; y2}{; distanceCouleurDébut ; distanceCouleurFin}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
id	Chaîne	→	Nom du dégradé
couleurDébut	Chaîne	→	Couleur de démarrage
couleurFin	Chaîne	→	Couleur de fin
rotation	Entier	→	Rotation du vecteur de dégradé
spreadMethod	Texte	→	Mode de diffusion (pad, reflect ou repeat)
x1	Réel	→	Coordonnée x1 du vecteur de dégradé (0 si omis)
y1	Réel	→	Coordonnée y1 du vecteur de dégradé (1 si omis)
x2	Réel	→	Coordonnée x2 du vecteur de dégradé (0 si omis)
y2	Réel	→	Coordonnée y2 du vecteur de dégradé (1 si omis)
distanceCouleurDébut	Réel	→	Distance de la couleur de démarrage (pourcentage)
distanceCouleurFin	Réel	→	Distance de la couleur de fin (pourcentage)
Résultat	Chaîne	←	Référence du dégradé

Description

La commande **SVG_Define_linear_gradient** définit un nouveau dégradé linéaire dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Un dégradé consiste en une transition de couleur progressive continue au long d'un vecteur, d'une couleur à l'autre. Une fois définis, les dégradés sont appelés sur un élément graphique donné. Vous devez indiquer si l'élément doit être rempli ou liseré avec le dégradé.

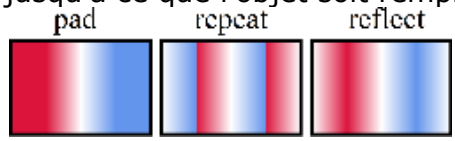
Le paramètre *id* spécifie le nom du dégradé. Si un élément de même nom existait, il est remplacé. Ce nom sera utilisé pour désigner le dégradé à chaque fois qu'une expression couleur sera attendue via la syntaxe "url(#ID)".

Les paramètres *couleurDébut* et *couleurFin* spécifient les couleurs utilisées pour débuter et terminer le dégradé.

Le paramètre optionnel *rotation* définit la position et le sens du vecteur de dégradé (cf. exemple).

Le paramètre optionnel *spreadMethod* permet de définir le remplissage lorsque le dégradé commence ou se termine à l'intérieur des limites de l'objet *objetSVGParent*. Vous pouvez passer dans ce paramètre l'une des chaînes suivantes :

- "pad" : utiliser les couleurs extrêmes du dégradé pour remplir le reste de la zone.
- "reflect" : refléter indéfiniment le motif du dégradé en début/fin puis fin/début puis début/fin, etc., jusqu'à ce que l'objet soit rempli.
- "repeat", répéter indéfiniment le motif du dégradé en début/fin, début/fin, début/fin, etc., jusqu'à ce que l'objet soit rempli.



Si ce paramètre est omis, l'effet de la valeur "pad" est utilisé.

Les paramètres optionnels *x1*, *y1*, *x2* et *y2* permettent de définir le vecteur du dégradé. Ce vecteur fournit des points de départ et d'arrivée utilisés par le moteur de rendu. Vous devez passer dans ces paramètres des pourcentages exprimés sous forme de ratios (0...1).

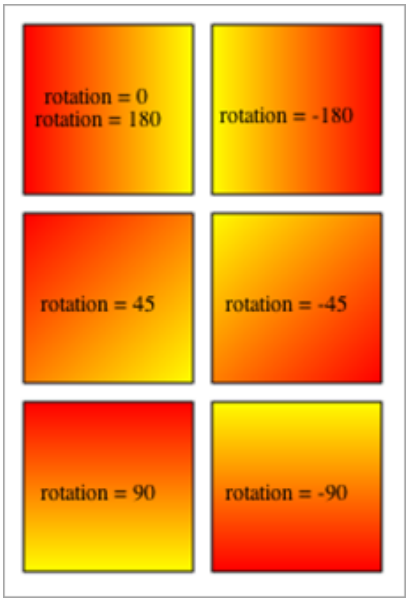
A compter de 4D v14, vous pouvez utiliser les paramètres optionnels *distanceCouleurDébut* et *distanceCouleurFin* pour définir la valeur en pourcentage, respectivement, de la distance entre le

point de départ de la couleur du dégradé et sa fin. Vous pouvez définir le pourcentage soit via un réel <1, soit via une valeur entre 0 et 100. Par exemple, les valeurs "0.1" et "10" seront toutes deux interprétées comme 10%.

Passer une valeur négative est interprété comme 0% pour le paramètre *distanceCouleurDébut* et 100% pour le paramètre *distanceCouleurFin*. Si vous passez une valeur >100, elle est interprétée comme 100%.

Exemple 1

Dessiner 6 carrés pleins utilisant chacun un serveur de peinture de dégradé linéaire en variant la rotation et le sens du vecteur de dégradé




```
$svg:=SVG_New

SVG_Define_linear_gradient($svg;"demoGradient_1";"red";"yellow")
SVG_New_rect($svg;10;10;90;90;0;0;"black";"url(#demoGradient_1)")
SVG_New_text($svg;"rotation = 0\rrotation = 180";50;40;"";-1;-1;Aligné au centre)

SVG_Define_linear_gradient($svg;"demoGradient_2";"red";"yellow";-180)
SVG_New_rect($svg;110;10;90;90;0;0;"black";"url(#demoGradient_2)")
SVG_New_text($svg;"rotation = -180";150;50;"";-1;-1;Aligné au centre)

SVG_Define_linear_gradient($svg;"demoGradient_3";"red";"yellow";45)
SVG_New_rect($svg;10;110;90;90;0;0;"black";"url(#demoGradient_3)")
SVG_New_text($svg;"rotation = 45";50;150;"";-1;-1;Aligné au centre)

SVG_Define_linear_gradient($svg;"demoGradient_4";"red";"yellow";-45)
SVG_New_rect($svg;110;110;90;90;0;0;"black";"url(#demoGradient_4)")
SVG_New_text($svg;"rotation = -45";150;150;"";-1;-1;Aligné au centre)

SVG_Define_linear_gradient($svg;"demoGradient_5";"red";"yellow";90)
SVG_New_rect($svg;10;210;90;90;0;0;"black";"url(#demoGradient_5)")
SVG_New_text($svg;"rotation = 90";50;250;"";-1;-1;Aligné au centre)

SVG_Define_linear_gradient($svg;"demoGradient_6";"red";"yellow";-90)
SVG_New_rect($svg;110;210;90;90;0;0;"black";"url(#demoGradient_6)")
SVG_New_text($svg;"rotation = -90";150;250;"";-1;-1;Aligné au centre)

 //Sauvegarder le document
```

```
SVG_SAVE_AS_TEXT($svg;"test.svg")
 //Libérer la mémoire
SVG_CLEAR($svg)
```

Exemple 2

Cet exemple utilise les paramètres *distanceCouleurDébut* et *distanceCouleurFin* (ajoutés dans 4D v14):

```
$Dom_node:=SVG_Define_linear_gradient($Dom_svg;"clicked";"black:50";"black:20";-90;"reflect";0;80)
```

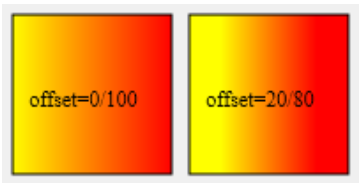
La définition suivante est ajoutée :

```
<linearGradient id="clicked spreadMethod="reflect x1="0" x2="0" y1="1" y2="0">      <stop offset="0%" stop-color="black" stop-opacity="0.5"/>      <stop offset="80%" stop-color="black" stop-opacity="0.2"/> </linearGradient>
```

Exemple 3

Cet exemple illustre l'effet des paramètres *distanceCouleurDébut* et *distanceCouleurFin* :

```
$svg:=SVG_New

SVG_Define_linear_gradient($svg;"demoGradient_1";"red";"yellow";-180;"reflect")
SVG_New_rect($svg;10;10;90;90;0;0;"black";"url(#demoGradient_1)")
SVG_New_text($svg;"offset=0/100";50;50;"";-1;-1;Aligné au centre)

SVG_Define_linear_gradient($svg;"demoGradient_2";"red";"yellow";-180;"reflect";20;80)
SVG_New_rect($svg;110;10;90;90;0;0;"black";"url(#demoGradient_2)")
SVG_New_text($svg;"offset=20/80";150;50;"";-1;-1;Aligné au centre)
  //Stocker le document
SVG_SAVE_AS_TEXT($svg;"test2.svg")
  //Libérer la mémoire
SVG_CLEAR($svg)
```



SVG_Define_marker

SVG_Define_marker (objetSVGParent ; id {; x ; y {; largeur ; hauteur {; orientation}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
id	Chaîne	➡	Nom du symbole
x	Entier long	➡	Coordonnée sur l'axe x du point de référence
y	Entier long	➡	Coordonnée sur l'axe y du point de référence
largeur	Entier long	➡	Largeur du marqueur
hauteur	Entier long	➡	Hauteur du marqueur
orientation	Entier long	➡	Orientation du marqueur
Résultat	Ref_SVG	↩	Référence du marqueur

Description

La commande *SVG_Define_marker* crée un marqueur dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Un objet marqueur est utilisé pour le dessin de flèches ou de marqueurs multiples (les points d'une courbe par exemple). Un marqueur sera attaché à un élément SVG avec la commande *SVG_SET_MARKER*.

Le paramètre *id* spécifie le nom du marqueur. Le nom sera utilisé pour associer un marqueur à un objet. Si un élément de même nom existait, il est remplacé.

Les paramètres optionnels *x* et *y* spécifient les coordonnées du point de référence qui doit s'aligner exactement sur la position du marqueur.

Les paramètres optionnels *largeur* et *hauteur* précisent la largeur et la hauteur du rectangle de rendu.

Le paramètre optionnel *orientation* permet de régler l'orientation du marqueur. Une valeur comprise entre 0 et 360 représente l'angle entre l'axe x du marqueur et celui de l'espace utilisateur. Si ce paramètre est omis ou si sa valeur n'est pas dans l'intervalle [0 - 360] le placement sera calculé automatiquement par le moteur de rendu en fonction de l'objet.

Exemple

Reportez-vous à l'exemple de la commande *SVG_SET_MARKER*.

SVG_Define_pattern

SVG_Define_pattern (objetSVGParent ; idMotif {; largeur {; hauteur {; x {; y {; unité {; viewBox}}}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
idMotif	Texte	➡	Nom du motif
largeur	Entier long	➡	Largeur du motif
hauteur	Entier long	➡	Hauteur du motif
x	Entier long	➡	Position x du motif
y	Entier long	➡	Position y du motif
unité	Texte	➡	Unité des longueurs et positions
viewBox	Texte	➡	Rectangle de visualisation
Résultat	Ref_SVG	↩	Référence du motif

Description

La commande *SVG_Define_pattern* définit un nouveau motif personnalisé dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas (ou n'appartient pas à) un document SVG, une erreur est générée.

Le paramètre *idMotif* spécifie le nom du motif. Le nom sera utilisé pour associer le motif à un objet. Si un élément de même nom existait, une erreur est générée.

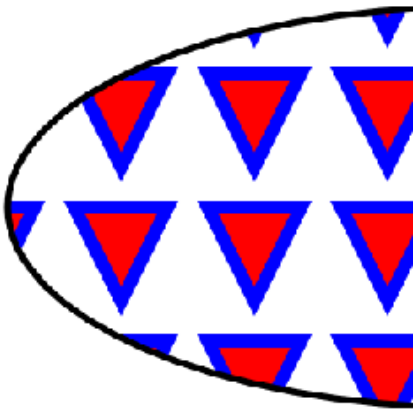
Les paramètres optionnels *largeur*, *hauteur*, *x*, *y*, *unité* et *viewBox* définissent le rectangle de référence du motif, c'est-à-dire la manière dont la mosaïque de motif sera placée et espacée.

Le motif sera associé comme peinture de remplissage ou de contour en passant la chaîne "url(#id)" comme valeur lorsqu'une expression couleur est attendue.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/pservers.html#Patterns

Exemple 1

Définition d'un motif et utilisation pour le remplissage d'une ellipse :

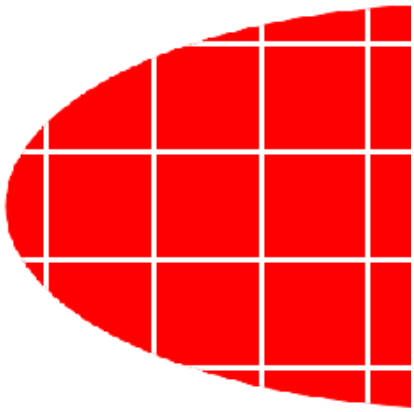


```
//Définition du pattern
$Dom_pattern:=SVG_Define_pattern($Dom_SVG;"MyPattern";100;100;0;0;"";"0 0 10 10")
$Dom_path:=SVG_New_path($Dom_pattern;0;0)

SVG_PATH_MOVE_TO($Dom_path;0;0)
SVG_PATH_LINE_TO($Dom_path;7;0)
SVG_PATH_LINE_TO($Dom_path;3,5;7)
SVG_PATH_CLOSE($Dom_path)
```

```
SVG_SET_FILL_BRUSH($Dom_path;"red")
SVG_SET_STROKE_BRUSH($Dom_path;"blue")

 //Tracé d'une ellipse remplie avec le motif
$Dom_ellipse:=SVG_New_ellipse($Dom_SVG;400;200;350;150;"black";"url(#MyPattern)";5)
```

Exemple 2

Définition d'un motif et utilisation pour le remplissage et les contours d'une ellipse :



```
 //Définition du motif
$Dom_pattern:=SVG_Define_pattern($Dom_SVG;"MyPattern ";80;80;0;0;"";"0 0 20 20")
$Dom_rect:=SVG_New_rect($Dom_pattern;0;0;20;20;0;0;"white";"red")

 //Tracé d'une ellipse
$Dom_ellipse:=SVG_New_ellipse($Dom_SVG;400;200;350;150)

 //Utiliser le motif pour le remplissage et les contours
SVG_SET_FILL_BRUSH($Dom_ellipse;"url(#MyPattern)")
SVG_SET_STROKE_BRUSH($Dom_ellipse;"url(#MyPattern)")
```

SVG_Define_radial_gradient

SVG_Define_radial_gradient (objetSVGParent ; id ; couleurDébut ; couleurFin {; cx ; cy ; r {; fx ; fy}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
id	Chaîne	➡	Nom du dégradé
couleurDébut	Chaîne	➡	Couleur de démarrage
couleurFin	Chaîne	➡	Couleur de fin
cx	Entier	➡	Coordonnée du centre de couleurFin sur l'axe x
cy	Entier	➡	Coordonnée du centre de couleurFin sur l'axe y
r	Entier	➡	Rayon de couleurFin
fx	Entier	➡	Coordonnée du centre de couleurDébut sur l'axe x
fy	Entier	➡	Coordonnée du centre de couleurDébut sur l'axe y
Résultat	Chaîne	↩	Référence du dégradé

Description

La commande *SVG_Define_radial_gradient* définit un nouveau dégradé radial dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Un dégradé consiste en une transition de couleur progressive continue au long d'un vecteur, d'une couleur à l'autre. Une fois définis, les dégradés sont appelés sur un élément graphique donné. Vous devez indiquer si l'élément doit être rempli ou liseré avec le dégradé.

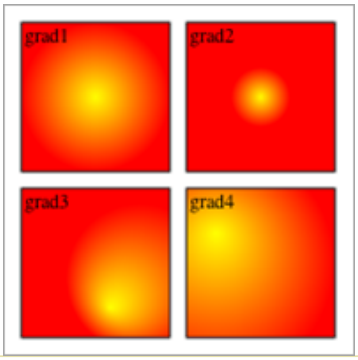
Le paramètre *id* spécifie le nom du dégradé. Si un élément de même nom existait, il est remplacé. Ce nom sera utilisé pour désigner le dégradé à chaque fois qu'une expression couleur sera attendue via la syntaxe "url(#ID)".

Les paramètres *couleurDébut* et *couleurFin* spécifient les couleurs utilisées pour débuter et terminer le dégradé.

Les paramètres optionnels *cx*, *cy* et *r* définissent, en pourcentage, le cercle limite externe de *couleurFin* du dégradé. Leurs valeurs doivent être comprises entre 0 et 100.

Les paramètres optionnels *fx* et *fy* définissent, en pourcentage, le foyer du dégradé. La *couleurDébut* commence au point [*fx,fy*]. Leurs valeurs doivent être comprises entre 0 et 100. Si ces arguments sont omis, ce point coïncide avec [*cx,cy*].

Exemple




```
$svg:=SVG_New

SVG_Define_radial_gradient($svg;"grad1";"yellow";"red")
SVG_New_rect($svg;10;10;90;90;0;0;"black";"url(#grad1)")
SVG_New_text($svg;"grad1";12;10)
```

```
SVG_Define_radial_gradient($svg;"grad2";"yellow";"red";50;50;20;50;50)
SVG_New_rect($svg;110;10;90;90;0;0;"black";"url(#grad2)")
SVG_New_text($svg;"grad2";112;10)

SVG_Define_radial_gradient($svg;"grad3";"yellow";"red";80;60;50;60;80)
SVG_New_rect($svg;10;110;90;90;0;0;"black";"url(#grad3)")
SVG_New_text($svg;"grad3";12;110)

SVG_Define_radial_gradient($svg;"grad4";"yellow";"red";20;50;80;20;30)
SVG_New_rect($svg;110;110;90;90;0;0;"black";"url(#grad4)")
SVG_New_text($svg;"grad4";112;110)

 `Sauvegarder le document
SVG_SAVE_AS_TEXT($svg;"test.svg")
 `Libérer la mémoire
SVG_CLEAR($svg)
```

SVG_Define_shadow

SVG_Define_shadow (objetSVGParent ; id {; déviation {; offsetX {; offsetY}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
id	Chaîne	➡	Nom du filtre
déviation	Entier long	➡	Valeur de dispersion de l'ombre
offsetX	Entier long	➡	Décalage sur l'axe x
offsetY	Entier long	➡	Décalage sur l'axe y
Résultat	Ref_SVG	⮌	Référence du filtre

Description

La commande *SVG_Define_shadow* définit un nouveau filtre d'ombrage dans le conteneur SVG désigné par *objetSVGParen*t et retourne sa référence. Si *objetSVGParen*t n'est pas un document SVG, une erreur est générée.

Un fois défini, un filtre est appliqué aux objets souhaités via la commande *SVG_SET_FILTER*.

Le paramètre *id* spécifie le nom du filtre. Le nom sera utilisé pour associer un filtre à un objet. Si un élément de même nom existait, il est remplacé.

Le paramètre optionnel *déviation* définit l'intensité de la dispersion de l'ombre. La valeur par défaut est 4.

Les paramètres optionnels *offsetX* et *offsetY* définissent respectivement le décalage horizontal et vertical de l'ombre par rapport à l'objet. La valeur par défaut est 4.

Exemple

Déclaration d'un filtre permettant de générer une ombre sous un texte :


```
$svg:=SVG_New

$text:=SVG_New_text($svg;"SVG";52;76-45;"Verdana";45)
SVG_SET_FONT_COLOR($text;"red")
 `Définir le filtre
SVG_Define_shadow($svg;"myShadow")
 `et l'appliquer au texte
SVG_SET_FILTER($text;"myShadow")
```

SVG_Define_solidColor

SVG_Define_solidColor (objetSVGParent ; id ; couleur {; opacité}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
id	Chaîne	➡	Nom de la couleur
couleur	Chaîne	➡	Expression couleur
opacité	Entier long	➡	Opacité
Résultat	Ref_SVG	⮌	Référence de la couleur

Description

La commande *SVG_Define_solidColor* définit une nouvelle couleur personalisée dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Le paramètre *id* spécifie le nom de la couleur. Le nom sera utilisé pour associer une couleur à un objet. Si un élément de même nom existait, il est remplacé.

Le paramètre *couleur* est une expression couleur reconnue par le SVG (cf. **Couleurs et dégradés**).

Le paramètre optionnel *opacité* permet de préciser une opacité (de 0 à 100) pour cette couleur. Si ce paramètre est omis, l'opacité est de 100 %.

Pour utiliser la couleur ainsi définie comme peinture de remplissage ou de contour, passez la chaîne "url(#id)" comme valeur lorsqu'une expression couleur est attendue.

Exemples

```
`Définir un bleu à 50 %
SVG_Define_solidColor($svg;"MaCouleur";"blue";50)

SVG_New_rect($svg;0;0;20;20;0;0;"url(#MaCouleur)";"url(#MaCouleur)")

$line:=SVG_New_line(10;10;100;100)
SVG_SET_STROKE_BRUSH($line;"url(#MaCouleur)")
```

SVG_Define_style

SVG_Define_style (objetSVGParent ; style {; type {; media}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
style	Texte	→	Définition du style OU Chemin d'accès du fichier à utiliser
type	Texte	→	Type de contenu
media	Texte	→	Descripteur de media
Résultat	Ref_SVG	←	Référence du style

Description

La commande *SVG_Define_style* définit une nouvelle feuille de style dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas (ou n'appartient pas à) un document SVG, une erreur est générée.

Le paramètre *style* permet l'incorporation de feuilles de styles directement dans un contenu SVG :

- Si le paramètre *style* contient un chemin d'accès valide à un fichier CSS, la définition du style est faite à l'aide du mécanisme de référencement de feuilles de style externes. Le chemin, s'il commence par le caractère # ou la chaîne " file : ", exprime un chemin relatif dont la racine est le dossier "Resources " de la base.
- Le paramètre *style* peut être un URL absolu du type "http://… ", dans ce cas la feuille de style sera référencée comme ressource externe.
- Vous pouvez également passer dans *style* un URL relatif au sous-dossier "Resources/SVG/" de la base. Cette possibilité est particulièrement utile en client/serveur, lorsque les fichiers sont stockés dans le dossier "Resources". Les URLs relatifs peuvent débuter par :
  - "**/**", désignant le chemin "~/Resources/SVG/"
  - "**./**", désignant le chemin "~/Resources/"
  - "**../**", désignant le dossier de la base

  Pour des exemples d'URL relatifs, reportez-vous à la commande **SVG_New_image**.

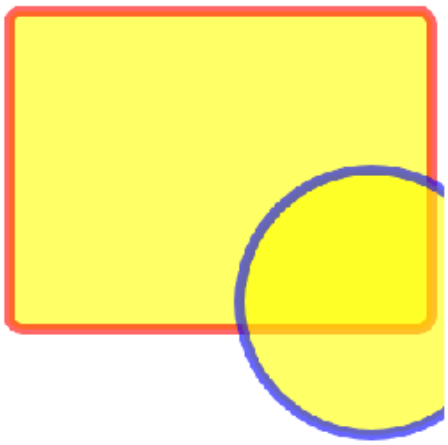
Le paramètre optionnel *type* spécifie le langage de la feuille de style du contenu de l'élément. La valeur par défaut est "text/css".

Le paramètre optionnel *media* indique le media de destination souhaité pour l'information de style. Si vous omettez ce paramètre, la valeur par défaut utilisée est "all". Si la valeur n'est pas comprise dans la liste des types de medias reconnus par CSS2, une erreur est générée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/styling.html#StyleElement

Exemple 1

Définition d'un style incorporé et surcharge d'un des éléments :



```
 //Définition du style
$Txt_style:=".colored {fill: yellow; fill-opacity: 0.6; stroke: red;stroke-width:8; stroke-opacity: 0.6}"
SVG_Define_style($Dom_SVG;$Txt_style)

 //Création d'un groupe et affectation d'un style par défaut
$Dom_g:=SVG_New_group($Dom_SVG)
SVG_SET_CLASS($Dom_g;"colored")

 //Tracé d'un rectangle
$Dom_rect:=SVG_New_rect($Dom_g;25;30;320;240;10;10;"";"")

 //Tracé d'un cercle et surcharge du style avec une couleur de contour personnalisée
$Dom_circle:=SVG_New_circle($Dom_g;300;250;100;"";"")
SVG_SET_STROKE_BRUSH($Dom_circle;"blue")
```

Exemple 2

Référencement du fichier "monstyle.css" placé dans le dossier "dev" du dossier "Resources" :

```
 //Définition du style
SVG_Define_style($Dom_svg;"#dev/monstyle.css")

 //Création d'un groupe et affectation d'un style par défaut
$Dom_g:=SVG_New_group($Dom_SVG)
SVG_SET_CLASS($Dom_g;"colored")

 //Tracé d'un rectangle
$Dom_rect:=SVG_New_rect($Dom_g;25;30;320;240;10;10;"";"")
```

Fichier monstyle.css :

```
.colored {fill: red; fill-opacity: 0.6; stroke: blue; stroke-width:8; stroke-opacity: 0.6}
```

SVG_DEFINE_STYLE_WITH_ARRAYS

SVG_DEFINE_STYLE_WITH_ARRAYS (objetSVG ; ptrTabNoms ; ptrTabValeurs {; className {; type {; media {; titre}}}})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'objet SVG
ptrTabNoms	Pointeur	➡	Pointeur vers le tableau des noms de styles
ptrTabValeurs	Pointeur	➡	Pointeur vers le tableau des valeurs de styles
className	Texte	➡	Nom de la classe du style CSS
type	Texte	➡	Type de contenu
media	Texte	➡	Descripteur de media
titre	Texte	➡	Nom du style

Description

La commande **SVG_DEFINE_STYLE_WITH_ARRAYS** définit les styles de l'objet SVG désigné par le paramètre *objetSVG* à l'aide de tableaux.

- Si le paramètre *objetSVG* désigne l'élément racine, les styles sont définis en tant qu'éléments "style" inclus dans la section "defs" (*Internal Style Sheet*). Dans ce cas, le paramètre *className* est obligatoire (s'il est manquant, une erreur est retournée). Vous pouvez ensuite affecter la feuille de style *className* à des objets SVG en passant son nom à la commande **SVG_SET_CLASS** (voir exemple 1).
- Si le paramètre *objetSVG* désigne un élément SVG autre que l'élément racine, le style est défini en tant qu'attribut de style pour cet élément (*Inline Style Sheet*) (voir exemple 2).

Le paramètre optionnel *type* spécifie le langage de la feuille de style du contenu de l'élément. La valeur par défaut est "text/css".

Le paramètre optionnel *media* indique le media de destination souhaité pour l'information de style. Si vous omettez ce paramètre, la valeur par défaut utilisée est "all". Si la valeur n'est pas comprise dans la liste des types de medias reconnus par CSS2, une erreur est générée.

Le paramètre optionnel *titre* vous permet d'ajouter un attribut de type "title".

Exemple 1

Exemple de définition de styles internes :

```
TABLEAU TEXTE($tnoms;0)
TABLEAU TEXTE($tvaleurs;0)
AJOUTER A TABLEAU($tnoms;"fill")
AJOUTER A TABLEAU($tvaleurs;"black")
AJOUTER A TABLEAU($tnoms;"font-family")
AJOUTER A TABLEAU($tvaleurs;"'Lucida Grande' Verdana")
AJOUTER A TABLEAU($tnoms;"font-size")
AJOUTER A TABLEAU($tvaleurs;"20px")
AJOUTER A TABLEAU($tnoms;"text-align")
AJOUTER A TABLEAU($tvaleurs;"center")

$svg:=SVG_New
SVG_DEFINE_STYLE_WITH_ARRAYS($svg;->$tnoms;->$tvaleurs;"title")
$object:=SVG_New_textArea($svg;"Hello World!";10;10;200;310)
SVG_SET_CLASS($object;"title")
```

Cette méthode génère le code suivant :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?> <svg
xmlns="http://www.w3.org/2000/svg">      <defs id="4D">          <style
type="text/css">.title{fill:red;font-family:'Lucida Grande' Verdana;font-size:20px;text-align:center;}
</style>      </defs>      <textArea class="title" height="310" width="200" x="10" y="10">Hello
World!</textArea> </svg>
```

Exemple 2

Exemple de définition de styles inline :

```
TABLEAU TEXTE($tnoms;0)
TABLEAU TEXTE($tvaleurs;0)
AJOUTER A TABLEAU($tnoms;"fill")
AJOUTER A TABLEAU($tvaleurs;"black")
AJOUTER A TABLEAU($tnoms;"font-family")
AJOUTER A TABLEAU($tvaleurs;"'Lucida Grande' Verdana")
AJOUTER A TABLEAU($tnoms;"font-size")
AJOUTER A TABLEAU($tvaleurs;"20px")
AJOUTER A TABLEAU($tnoms;"text-align")
AJOUTER A TABLEAU($tvaleurs;"center")

$svg:=SVG_New
$object:=SVG_New_textArea($svg;"Hello World!";10;10;200;310)
SVG_DEFINE_STYLE_WITH_ARRAYS($object;->$tnoms;->$tvaleurs)
```

Cette méthode génère le code suivant :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?> <svg
xmlns="http://www.w3.org/2000/svg">      <textArea height="310" style="fill:red;font-
family:'Lucida Grande' Verdana;font-size:20px;text-align:center;" width="200" x="10" y="10">Hello
World!</textArea> </svg>
```

SVG_Define_symbol

SVG_Define_symbol (objetSVGParent ; id {; x {; y {; largeur {; hauteur {; mode}}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
id	Chaîne	→	Nom du symbole
x	Entier long	→	Position X du rectangle de visualisation
y	Entier long	→	Position Y du rectangle de visualisation
largeur	Entier long	→	Largeur du rectangle de visualisation
hauteur	Entier long	→	Hauteur du rectangle de visualisation
mode	Chaîne	→	Adaptation au rectangle de visualisation
Résultat	Ref_SVG	←	Référence du symbole

Description

La commande *SVG_Define_symbol* crée un symbole dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Un objet symbole est utilisé pour définir des objets graphiques qui pourront être instanciés en utilisant la commande *SVG_Use*.

Le paramètre *id* spécifie le nom du symbole.

Les paramètres optionnels *x*, *y*, *largeur* et *hauteur* spécifient le rectangle de la zone de visualisation ('viewBox').

Le paramètre optionnel *mode* permet d'indiquer si le graphique doit s'adapter, et comment, à la taille du rectangle de visualisation. Pour plus d'informations sur ce point, reportez-vous à la description de la commande *SVG_New*.

Exemple

Reportez-vous à la description de la commande *SVG_Use*.

SVG_DELETE_OBJECT

SVG_DELETE_OBJECT (objetSVG)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇨	Référence d’un élément SVG

Description

La commande *SVG_DELETE_OBJECT* supprime l’objet SVG désigné par *objetSVG* du document auquel il appartient. Une erreur est générée si *objetSVG* n’est pas une référence valide.

SVG_Get_default_encoding

SVG_Get_default_encoding -> Résultat

Paramètre	Type		Description
Résultat	Texte		Jeu de caractères

Description

La commande *SVG_Get_default_encoding* retourne l'encodage utilisé lors de la création d’un nouveau document.

SVG_New_group

SVG_New_group (objetSVGParent {; id {; url {; cible}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
id	Chaîne	➡	Nom du groupe
url	Chaîne	➡	Lien externe
cible	Chaîne	➡	Cible du lien
Résultat	Ref_SVG	↩	Référence du groupe

Description

La commande *SVG_New_group* crée un groupe dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un groupe ou un document SVG valide, une erreur est générée.

Un groupe (élément 'g') permet de regrouper plusieurs éléments graphiques reliés, qui hériteront des propriétés du groupe.

Le paramètre optionnel *id* permet d'attribuer un nom au groupe. Les groupes nommés sont nécessaires pour plusieurs finalités telles que l'animation et les objets réutilisables.

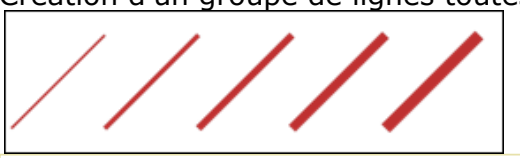
Le paramètre optionnel *url* permet d'associer un lien externe. Les objets du groupe sont alors cliquables (analogue à l'élément 'a' du HTML).

Le paramètre optionnel *cible* spécifie le nom de la cible dans laquelle le document doit s'ouvrir quand le lien est activé. Les valeurs attendues sont celles du HTML auxquelles s'ajoutent la chaîne "new" pour l'ouverture dans une nouvelle fenêtre et "none" qui équivaut à ne pas traiter cet attribut.

Note : Les liens externes sont ignorés lorsque le SVG est affiché dans un objet image (variable ou champ) d'un formulaire 4D. La gestion des références externes est effectuée par le moteur de rendu. Dans ces conditions, le résultat peut dépendre de la plate-forme et du logiciel de visualisation.

Exemple 1

Création d'un groupe de lignes toutes de la même couleur :



```
$SVG:=SVG_New
$group:=SVG_New_group($SVG)
 `Attribuer une couleur aux éléments du groupe
SVG_SET_STROKE_BRUSH($group;"firebrick")
$newobject:=SVG_New_line($group;100;300;300;100;"";5)
$newobject:=SVG_New_line($group;300;300;500;100;"";10)
$newobject:=SVG_New_line($group;500;300;700;100;"";15)
$newobject:=SVG_New_line($group;700;300;900;100;"";20)
$newobject:=SVG_New_line($group;900;300;1100;100;"";25)
```

Exemple 2

Création d'un texte cliquable :


```
$SVG:=SVG_New
$group:=SVG_New_group($SVG;"w3Link";"http://www.w3.org";"new")
$newobject:=SVG_New_text($group;"www.w3.org";10;10;"arial";12;Souligné;Aligné à gauche;"blue")
```

SVG_SET_DEFAULT_ENCODING

SVG_SET_DEFAULT_ENCODING {(encodage)}

Paramètre	Type		Description
encodage	Chaîne	⇨	Jeu de caractères

Description

La commande *SVG_SET_DEFAULT_ENCODING* permet de fixer l'encodage qui sera utilisé lors des prochaines créations de documents.

Si le paramètre *encodage* est omis, la commande rétablit le jeu de caractères par défaut : "UTF-8".

SVG_Set_description

SVG_Set_description (objetSVGParent ; description) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
description	Chaîne	➡	Texte du commentaire
Résultat	Ref_SVG	↩	Référence de la description

Description

La commande *SVG_Set_description* définit un texte pour l'élément SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un élément SVG, une erreur est générée.

Une description est généralement utilisée pour insérer un commentaire ou un texte explicatif dans le code SVG.

Exemple

```
$SVG:=SVG_New
$g:=SVG_group($SVG)
SVG_Set_title($g;"Ventes de la société par région")
SVG_Set_description($g;"Diagramme en barre des ventes de la société par région.")
...
```

SVG_SET_PATTERN_CONTENT_UNITS

SVG_SET_PATTERN_CONTENT_UNITS (motifObjet ; sysCoordonnées)

Paramètre	Type		Description
motifObjet	Ref_SVG	⇒	Référence du motif à modifier
sysCoordonnées	Texte	⇒	Système de coordonnées à utiliser

Description

La commande *SVG_SET_PATTERN_CONTENT_UNITS* définit le système de coordonnées pour le contenu du motif désigné par *motifObjet*. Si *motifObjet* n'est pas un motif, une erreur est générée.

Le paramètre *sysCoordonnées* spécifie le nom du système à utiliser. Il doit être égal à "userSpaceOnUse" ou "objectBoundingBox" sinon une erreur est générée.

Référence :
http://www.yoyodesign.org/doc/w3c/svg1/pservers.html#PatternContentUnitsAttribute

SVG_SET_PATTERN_UNITS

SVG_SET_PATTERN_UNITS (motifObjet ; sysCoordonnées)

Paramètre	Type		Description
motifObjet	Ref_SVG	⇒	Référence du motif à modifier
sysCoordonnées	Texte	⇒	Système de coordonnées à utiliser

Description

La commande *SVG_SET_PATTERN_UNITS* définit le système de coordonnées pour les attributs *x*, *y*, *largeur* et *hauteur* du motif désigné par *motifObjet*. Si *motifObjet* n'est pas un motif, une erreur est générée.

Le paramètre *sysCoordonnées* spécifie le nom du système à utiliser. Il doit être égal à "userSpaceOnUse" ou "objectBoundingBox" sinon une erreur est générée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/pservers.html#PatternUnitsAttribute

SVG_Set_title

SVG_Set_title (objetSVGParent ; titre) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	⇒	Référence de l'élément parent
titre	Chaîne	⇒	Texte du titre
Résultat	Ref_SVG	⇐	Référence du titre

Description

La commande *SVG_Set_title* définit un titre pour l'élément SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un élément SVG, une erreur est générée.

Un titre est une donnée textuelle qui n'est pas incluse dans le rendu de l'image mais est utile pour structurer des documents complexes. Certains moteurs de rendu SVG utilisent le texte de cet élément pour afficher une infobulle au moment du survol de l'objet.

Exemple

```
$SVG:=SVG_New$rec:=SVG_New_rect($SVG;20;20;650;650;0;0;"gray";"lemonchiffon")
SVG_Set_title($rec;"Rectangle de fond")
$Symbol:=SVG_Define_symbol($SVG;"MySymbol";0;0;110;110;"true")
SVG_Set_title($Symbol;"Définition d'un symbole comprenant 2 carrés et 2 cercles ")
...
```

Texte

- SVG_APPEND_TEXT_TO_TEXTAREA
- SVG_Get_text
- SVG_New_text
- SVG_New_textArea
- SVG_New_tspan
- SVG_New_vertical_text
- SVG_SET_FONT_COLOR
- SVG_SET_FONT_FAMILY
- SVG_SET_FONT_SIZE
- SVG_SET_FONT_STYLE
- SVG_SET_TEXT_ANCHOR
- SVG_SET_TEXT_KERNING
- SVG_SET_TEXT_LETTER_SPACING
- SVG_SET_TEXT_RENDERING
- SVG_SET_TEXT_WRITING_MODE
- SVG_SET_TEXTAREA_TEXT

SVG_APPEND_TEXT_TO_TEXTAREA

SVG_APPEND_TEXT_TO_TEXTAREA (objetSVG ; texteAjout)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément texte
texteAjout	Texte	➡	Texte à ajouter

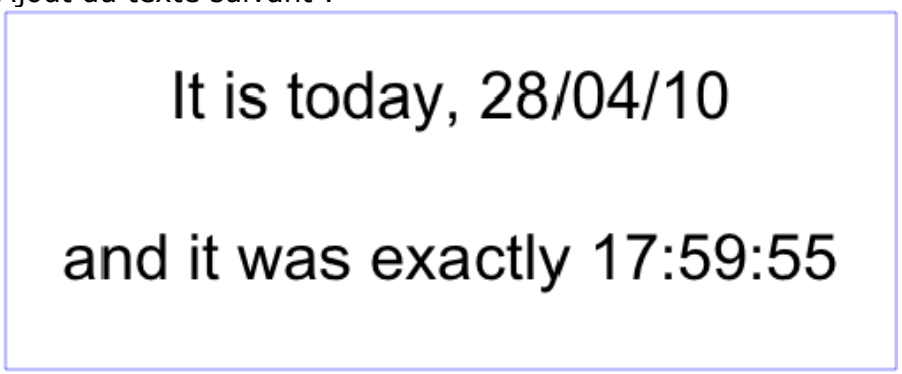
Description

La commande *SVG_APPEND_TEXT_TO_TEXTAREA* permet d'ajouter du texte au contenu textuel de l'objet texte désigné par *objetSVG*. Si *objetSVG* n'est pas un objet "textArea", une erreur est générée.

Les caractères retour à la ligne sont automatiquement remplacés par des éléments "<tbreak/>".

Exemple

Ajout du texte suivant :




```
//Afficher les contours à l'aide de l'élément 'rect'
$Dom_rect:=SVG_New_rect($Dom_SVG;10;10;500;200;0;0;"blue:50";"none")

//Créer le texte
$Dom_text:=SVG_New_textArea($Dom_SVG;"It is today, ";10;30;500;200;"'Arial'";36;0;3)

//Ajout de la date et de 2 RC
SVG_APPEND_TEXT_TO_TEXTAREA($Dom_text;Chaine(Date du jour)+"\r\r")

//Enfin, ajout de l'heure courante
SVG_APPEND_TEXT_TO_TEXTAREA($Dom_text;"and it was exactly "+Chaine(Heure courante))
```

SVG_Get_text

SVG_Get_text (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d’un élément texte
Résultat	Texte	⇐	Contenu de texte

Description

La commande *SVG_Get_text* retourne le contenu textuel de l’élément désigné par *objetSVG*. Si *objetSVG* n’est pas une référence d’objet texte ('text', 'textArea' ou 'tspan'), une erreur est générée.

Dans le cas d’un objet 'textArea', les éléments <tbreak/> sont convertis en CR.

SVG_New_text

SVG_New_text (objetSVGParent ; texte {; x {; y {; police | stylePolice {; taille {; style {; alignement {; couleur {; rotation {; interligne {; étirement}}}}}}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
texte	Texte	→	Texte à insérer
x	Réel	→	Coordonnée sur l'axe x
y	Réel	→	Coordonnée sur l'axe y
police | stylePolice	Texte	→	Nom de la police ou Définition de style
taille	Entier long	→	Taille des caractères en points
style	Entier long	→	Style des caractères
alignement	Entier long	→	Alignement
couleur	Chaîne	→	Couleur du texte
rotation	Réel	→	Angle de rotation du texte
interligne	Réel	→	Interlignage en point
étirement	Réel	→	Facteur d'étirement horizontal
Résultat	Ref_SVG	↩	Référence de l'objet texte SVG

Description

La commande *SVG_New_text* insère le *texte* dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Note : A compter de 4D v15, la commande **SVG_New_text** prend en charge le texte multi-style simple (texte pouvant contenir des styles différents, mais sans attributs SPAN imbriqués). Voir exemple 5.

Les paramètres optionnels *x* et *y* permettent de préciser le positionnement sur l'axe x et sur l'axe y du coin supérieur du premier caractère du *texte*. Ce point est différemment situé selon la valeur de l'alignement : à gauche pour un alignement à gauche, à droite pour un alignement à droite ou au centre lorsque le texte est centré.

La commande **SVG_New_text** accepte deux syntaxes différentes pour la définition des caractères :

- vous pouvez passer diverses valeurs dans les paramètres *police*, *taille*, *style*, *alignement*, *couleur*, *rotation*, *interligne* et *étirement* : *police* et *taille* permettent de spécifier la police et la taille, en points, à utiliser. Si ces paramètres sont omis, le texte sera écrit en **Times New Roman 12 pts**.
  Le paramètre optionnel *style* précise le style de caractères à utiliser. Vous devez passer dans le paramètre *style* l'une des valeurs suivantes ou une combinaison de ces valeurs (vous pouvez également utiliser les constantes 4D correspondantes dans le thème **Styles de caractères**) :

  0 = Normal
  1 = Gras
  2 = Italique
  4 = Souligné
  8 = Barré

  Le paramètre optionnel *alignement* permet de spécifier le type d'alignement appliqué au texte dessiné. Vous devez passer une des valeurs suivantes :

  2 = Aligné à gauche
  3 = Centré
  4 = Aligné à droite

Le paramètre optionnel *couleur* contient le nom de la couleur de la police. Pour plus d'informations sur les couleurs, reportez-vous au chapitre "**Couleurs et dégradés**".
Le paramètre optionnel *rotation* permet de préciser la rotation à appliquer au texte.
Le paramètre optionnel *interligne* permet de préciser la valeur de l'interlignage si le texte comporte plusieurs lignes. Valeur par défaut = 1.
Le paramètre optionnel *étirement* permet d'appliquer horizontalement un facteur d'étirement (valeur >1) ou de condensation (valeur comprise entre 0 et 1) au texte.

- ou bien, vous pouvez passer une définition de style dans le paramètre *défStyle* (en lieu et place du paramètre *police*) et omettre les paramètres suivants. Vous pouvez passer par exemple :

```
SVG_New_text($Dom_svg;"Hello World !";x;y;style_definition)
```

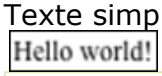
... où le paramètre *style_definition* contient une définition de style complète. Si vous passez par exemple "{font-size:48px;fill:red;}", cette définition sera ajoutée en tant qu'attribut de style sous la forme :

```
style="font-size:48px;fill:red;"
```

Dans ce cas, les éventuels paramètres suivants sont ignorés.

Exemple 1

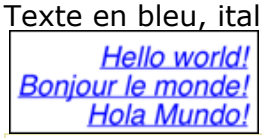
Texte simple utilisant les propriétés de texte par défaut :




```
$SVG:=SVG_New
$textID:=SVG_New_text($SVG;"Hello world!")
```

Exemple 2

Texte en bleu, italique, souligné et aligné à droite :




```
$SVG:=SVG_New
$text:="Hello world!\rBonjour le monde!\rHola Mundo!"
$size:=48
$font:="helvetica"
$textID:=SVG_New_text($SVG;$text;400;10;$font;$size;Italique+Souligné;Aligné à droite;"blue")
```

Exemple 3

Texte vertical :

```
$SVG:=SVG_New
$textID:=SVG_New_text($SVG;$text;-250;0;"";48;-1;-1;"red";-90)
```

Exemple 4

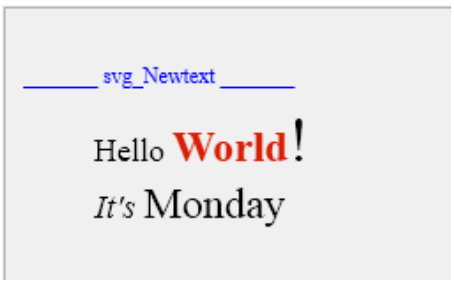
Texte condensé ou étiré :


```
$SVG:=SVG_New
$textID:=SVG_New_text($SVG;"Hello world (condensed)";0;0;"";-1;-1;-1;"blue";0;1;0,8)
$textID:=SVG_New_text($SVG;"Hello world (normal)";0;24)
$textID:=SVG_New_text($SVG;"Hello world (stretched)";0;48;"";-1;-1;-1;"red";0;1;2)
```

Exemple 5

Affichage de texte multistyle :

```
C_TEXTE($Dom_svg;$Dom_text;$Txt_buffer)
    //définition de texte mutlistyle
$Txt_buffer:="<SPAN STYLE=\"font-size:18pt\">Hello </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt;font-weight:bold;color:#D81E05\">World</SPAN>"+\
"<SPAN STYLE=\"font-size:36pt\">!</SPAN><BR/>"+\
"<SPAN STYLE=\"font-size:19pt;font-style:italic\">It's </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt\">Monday</SPAN>"
$Dom_svg:=SVG_New

  //titre
SVG_SET_FONT_COLOR(SVG_New_text($Dom_svg;"_______ svg_Newtext
_______";10;30);"blue")
  //texte
$Dom_text:=SVG_New_text($Dom_svg;$Txt_buffer;50;50)

SVGTool_SHOW_IN_VIEWER($Dom_svg)
SVG_CLEAR($Dom_svg)
```



SVG_New_textArea

SVG_New_textArea (objetSVGParent ; texte {; x {; y {; largeur {; hauteur {; police | stylePolice {; taille {; style {; alignement}}}}}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
texte	Texte	→	Texte à insérer
x	Entier long	→	Coordonnée sur l'axe x
y	Entier long	→	Coordonnée sur l'axe y
largeur	Entier long	→	Largeur de la zone de texte
hauteur	Entier long	→	Hauteur de la zone de texte
police | stylePolice	Texte	→	Nom de la police ou Définition de style
taille	Entier	→	Taille des caractères en points
style	Entier	→	Style des caractères
alignement	Entier	→	Alignement
Résultat	Ref_SVG	←	Référence de l'objet texte SVG

Description

La commande **SVG_New_textArea** insère une zone de texte dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

L'élément "textArea" est recommandé par la norme SVG tiny 1.2 et implémenté dans 4D v11 SQL à partir de la version 11.3 (cf. http://www.w3.org/TR/SVGMobile12/text.html#TextAreaElement). Cet élément implémente une zone de texte qui, au contraire de l'élément "text", gère automatiquement les retours à la ligne lorsque le texte dépasse la largeur demandée.

Notes :

- Dans l'élément "textArea", les retours à la ligne sont remplacés par des éléments "<tbreak/>".
- A compter de 4D v15, la commande **SVG_New_textArea** prend en charge le texte multi-style simple (texte pouvant contenir des styles différents, mais sans attributs SPAN imbriqués). Voir exemple 2.

Les paramètres optionnels *x* et *y* permettent de préciser le positionnement sur l'axe x et sur l'axe y du coin supérieur gauche de la zone.

Les paramètres optionnels *largeur* et *hauteur* définissent dans l'espace de coordonnées utilisateur la taille de la zone. Si l'un ou l'autre de ces paramètres n'est pas fourni, la zone de texte s'adaptera automatiquement à son contenu.

La commande **SVG_New_textArea** accepte deux syntaxes différentes pour la définition des caractères :

- vous pouvez passer diverses valeurs dans les paramètres *police*, *taille*, *style* et *alignement* : *police* et *taille* permettent de spécifier la police et la taille, en points, à utiliser. Si ces paramètres sont omis, le texte sera écrit en **Times New Roman 12 pts**.
  Le paramètre optionnel *style* précise le style de caractères à utiliser. Vous devez passer dans le paramètre *style* l'une des valeurs suivantes ou une combinaison de ces valeurs (vous pouvez également utiliser les constantes 4D correspondantes dans le thème **Styles de caractères**) :

  0 = Normal
  1 = Gras
  2 = Italique
  4 = Souligné
  8 = Barré

Le paramètre optionnel *alignement* permet de spécifier le type d'alignement appliqué au texte dessiné. Vous devez passer une des valeurs suivantes :

1 = Alignement par défaut (gauche)
2 = Aligné à gauche
3 = Centré
4 = Aligné à droite
5 = Justifié

- ou bien, vous pouvez passer une définition de style dans le paramètre *défStyle* (en lieu et place du paramètre *police*) et omettre les paramètres suivants. Vous pouvez passer par exemple :

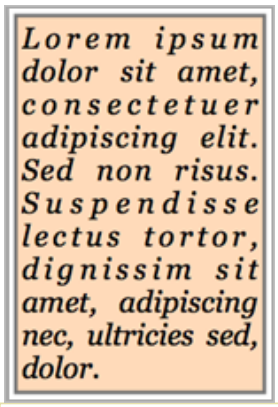
```
SVG_New_textArea($Dom_svg;"Hello World !";x;y;larg;haut;style_definition)
```

... où le paramètre *style_definition* contient une définition de style complète. Si vous passez par exemple "{font-size:48px;fill:red;}", cette définition sera ajoutée en tant qu'attribut de style sous la forme :

```
style="font-size:48px;fill:red;"
```

Dans ce cas, les éventuels paramètres suivants sont ignorés.

Exemple 1




```
$svg:=SVG_New
  `Positionner un rectangle de bordure
$rec:=SVG_New_rect($svg;5;5;210;320;0;0;"#777";"peachpuff";3)
  `Le texte
$txt:="Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor."
$txtArea:=SVG_New_textArea($svg;$txt;10;10;200;310;"Georgia";25;Italique;5)
  `Sauvegarder le document
SVG_SAVE_AS_TEXT($svg;"test.svg")
```

Exemple 2

Affichage de texte multistyle :

```
C_TEXTE($Dom_svg;$Dom_text;$Txt_buffer)
    //définition de texte mutlistyle
```

```
$Txt_buffer:="<SPAN STYLE=\"font-size:18pt\">Hello </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt;font-weight:bold;color:#D81E05\">World</SPAN>"+\
"<SPAN STYLE=\"font-size:36pt\">!</SPAN><BR/>"+\
"<SPAN STYLE=\"font-size:19pt;font-style:italic\">It's </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt\">Monday</SPAN>"
$Dom_svg:=SVG_New

   //titre
SVG_SET_FONT_COLOR(SVG_New_text($Dom_svg;"_______ SVG_New_textArea _______";10;30;"";-1);"blue")
   //textArea
$Dom_text:=SVG_New_textArea($Dom_svg;$Txt_buffer;50;50)

SVGTool_SHOW_IN_VIEWER($Dom_svg)
SVG_CLEAR($Dom_svg)
```

SVG_New_tspan

SVG_New_tspan (objetSVGParent ; texte {; x {; y {; police | défStyle {; taille {; style {; alignement {; couleur}}}}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	➡	Référence de l'élément parent
texte	Texte	➡	Texte à insérer
x	Entier long	➡	Coordonnée sur l'axe x
y	Entier long	➡	Coordonnée sur l'axe y
police | défStyle	Texte	➡	Nom de la police ou Définition de style
taille	Entier	➡	Taille des caractères en points
style	Entier	➡	Style des caractères
alignement	Entier	➡	Alignement
couleur	Chaîne	➡	Couleur du texte
Résultat	Ref_SVG	↩	Référence de l'objet texte SVG

Description

La commande **SVG_New_tspan** crée un nouvel élément dans l'élément 'text', 'tspan' ou 'textArea' désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas une référence d'un élément 'text', 'tspan' ou 'textArea', une erreur est générée.

Les différents paramètres optionnels sont décrits avec la commande **SVG_New_text**. Si certains paramètres optionnels sont omis, leurs valeurs sont héritées du ou des élément(s) parent(s).

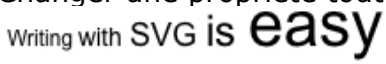
Exemple 1

Dans un texte il est possible de créer des paragraphes qui héritent des propriétés du parent.


```
$SVG:=SVG_New
 `Créer un nouveau texte en Arial, bleu, Aligné à gauche
$textID:=SVG_New_text($SVG;"";0;0;"arial";-1;-1;Aligné à gauche;"blue")
 `Paragraphes imbriqués avec indentation et changement de taille et de style
$textID:=SVG_New_tspan($textID;"TITRE 1";10;10;"";24;Gras+Souligné)
$textID:=SVG_New_tspan($textID;"Titre 2";20;42;"";12;Gras)
$textID:=SVG_New_tspan($textID;"Titre 3";30;60;"";10;Gras+Italique)
$textID:=SVG_New_tspan($textID;"Titre 4";40;78;"";8;Italique)
```

Exemple 2

Changer une propriété tout en restant dans un élément "text", ici la taille du texte :



```
$textID:=SVG_New_text($SVG;"Writing ";10;10;"arial";12)
SVG_SET_FONT_SIZE(SVG_New_tspan($textID;"with ");14)
SVG_SET_FONT_SIZE(SVG_New_tspan($textID;"SVG ");18)
SVG_SET_FONT_SIZE(SVG_New_tspan($textID;"is ");24)
SVG_SET_FONT_SIZE(SVG_New_tspan($textID;"easy ");36)
```

SVG_New_vertical_text

SVG_New_vertical_text (objetSVGParent ; texte {; x {; y {; police {; taille {; style {; alignement {; couleur {; rotation}}}}}}}}) -> Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
texte	Texte	→	Texte à insérer
x	Entier long	→	Coordonnée sur l'axe x
y	Entier long	→	Coordonnée sur l'axe y
police	Chaîne	→	Nom de la police
taille	Entier	→	Taille des caractères en points
style	Entier	→	Style des caractères
alignement	Entier	→	Alignement
couleur	Chaîne	→	Couleur du texte
rotation	Entier long	→	Angle de rotation du texte
Résultat	Ref_SVG	←	Référence de l'objet texte VG

Description

La commande *SVG_New_vertical_text* insère le *texte* verticalement dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas un document SVG, une erreur est générée.

Les paramètres optionnels *x* et *y* permettent de préciser le positionnement sur l'axe x et sur l'axe y du coin inférieur du premier caractère du *texte*.

Les paramètres optionnels *police* et *taille* permettent de spécifier la police et la taille, en points, à utiliser. Lorsque ces paramètres sont omis, le texte est écrit en Times New Roman 12 pts.

Le paramètre optionnel *style* précise le style de caractères à utiliser. Vous devez passer dans le paramètre *style* l'une des valeurs suivantes ou une combinaison de ces valeurs (vous pouvez également utiliser les constantes 4D correspondantes dans le thème "**Styles de caractères**") :

0 = Normal
1 = Gras
2 = Italique
4 = Souligné
8 = Barré

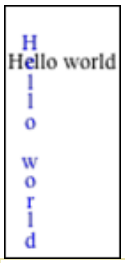
Le paramètre optionnel *alignement* permet de spécifier le type d'alignement appliqué au texte dessiné. Vous devez passer une des valeurs suivantes (vous pouvez également utiliser les constantes 4D correspondantes dans le thème "**Alignement objet**") :

2 = Aligné à gauche
3 = Centré
4 = Aligné à droite

Le paramètre optionnel *couleur* contient le nom de la couleur de la police. Pour plus d'informations sur les couleurs, reportez-vous au chapitre "**Couleurs et dégradés**".

Le paramètre optionnel *rotation* permet de préciser la rotation à appliquer au texte.

Exemple




```
$SVG:=SVG_New
$textID:=SVG_New_text($SVG;"Hello world";10;12)
$textID:=SVG_New_vertical_text($SVG;"Hello world";22;3;"";-1;-1;Centré;"blue")
```

SVG_SET_FONT_COLOR

SVG_SET_FONT_COLOR (objetSVG ; couleur)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d’un élément SVG
couleur	Chaîne	⇒	Couleur du texte

Description

La commande *SVG_SET_FONT_COLOR* permet de spécifier la couleur de la police pour l’objet SVG de référence *objetSVG*. Si *objetSVG* ne référence pas un élément valide, une erreur est générée.

Le paramètre *couleur* contient le nom de la couleur à utiliser. Pour plus d’informations sur les couleurs, reportez-vous à la section "**Couleurs SVG**".

SVG_SET_FONT_FAMILY

SVG_SET_FONT_FAMILY (objetSVG ; police {; police2 ; ... ; policeN})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément SVG
police	Chaîne	➡	Nom de police

Description

La commande *SVG_SET_FONT_FAMILY* permet de spécifier la police pour l'objet SVG de référence *objetSVG*.
Si *objetSVG* ne référence pas un élément valide, une erreur est générée.

Le paramètre *police* contient un nom de police à utiliser. Lorsque plusieurs noms sont passés, la commande construit automatiquement la liste des polices et/ou des familles génériques.

Exemple

Passage de plusieurs noms de police :

```
SVG_SET_FONT_FAMILY(objetSVG;"Lucida grande";"Sans-serif")
 // construira la liste : " 'Lucida grande' 'Sans-serif'"
```

SVG_SET_FONT_SIZE

SVG_SET_FONT_SIZ E (objetSVG ; taille)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d’un élément SVG
taille	Entier	⇒	Taille des caractères en points

Description

La commande *SVG_SET_FONT_SIZE* permet de spécifier la *taille* de la police pour l’objet SVG de référence *objetSVG*. Si *objetSVG* ne référence pas un élément valide, une erreur est générée.

Le paramètre *taille* contient la taille de la police exprimée en points.

SVG_SET_FONT_STYLE

SVG_SET_FONT_STYLE (objetSVG ; style)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d’un élément SVG
style	Entier	⇒	Style de caractères

Description

La commande *SVG_SET_FONT_STYLE* permet de spécifier le style du texte pour l’objet SVG de référence *objetSVG*. Si *objetSVG* ne référence pas un élément valide, une erreur est générée.

Vous devez passer dans le paramètre *style* l'une des valeurs suivantes ou la somme de plusieurs de ces valeurs :

0 = Normal
1 = Gras
2 = Italique
4 = Souligné
8 = Barré

SVG_SET_TEXT_ANCHOR

SVG_SET_TEXT_ANCHOR (objetSVG ; alignement)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d’un élément SVG
alignement	Entier	⇒	Alignement

Description

La commande *SVG_SET_TEXT_ANCHOR* permet de modifier l'alignement de l’objet SVG de référence *objetSVG*. Si *objetSVG* ne référence pas un élément valide, une erreur est générée.

Vous devez passer dans le paramètre *alignement* l'une des valeurs suivantes :

1 = Alignement par défaut (gauche)
2 = Aligné à gauche
3 = Centré
4 = Aligné à droite
5 = Justifié (pour un objet textArea seulement)

SVG_SET_TEXT_KERNING

SVG_SET_TEXT_KERNING (objetSVG ; crénage {; unité})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément texte
crénage	Réel	➡	Espacement des lettres
unité	Texte	➡	Unité de la valeur d'espacement

Description

La commande *SVG_SET_TEXT_KERNING* permet de modifier l'espacement entre les caractères (le crénage) de l'objet texte désigné par *objetSVG*. Si *objetSVG* n'est pas un objet texte SVG, une erreur est générée.

Le paramètre optionnel *unité* permet de préciser l'unité de la valeur d'espacement. La valeur par défaut est "%".

Si *crénage* vaut -1, la valeur d'espacement est fixée sur 'auto'.

Note : Sous Windows, l'implémentation est limitée au texte de gauche à droite et de haut en bas (désactivée pour le texte de droite à gauche) et aux éléments 'text' et 'tspan' ; sous Mac OS, la prise en charge n'est pas limitée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/text.html#KerningProperty

Exemple

Exemples de variation de crénage :


```
 //Référence
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;40;"";36)

$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_KERNING($Dom_text;0,5)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;120;"";36)
SVG_SET_TEXT_KERNING($Dom_text;1)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_KERNING($Dom_text;1,5)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_KERNING($Dom_text;2)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_KERNING($Dom_text;1,5)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;280;"";36)
```

```
SVG_SET_TEXT_KERNING($Dom_text;1)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_KERNING($Dom_text;0,5)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;360;"";36)
SVG_SET_TEXT_KERNING($Dom_text;0)
```

SVG_SET_TEXT_LETTER_SPACING

SVG_SET_TEXT_LETTER_SPACING (objetSVG ; espacement {; unité})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément texte
espacement	Réel	➡	Espacement des lettres
unité	Texte	➡	Unité de la valeur d'espacement

Description

La commande *SVG_SET_TEXT_LETTER_SPACING* permet de modifier l'espacement des lettres de l'objet texte désigné par *objetSVG* en plus de tout espacement dû à la propriété 'kerning'. Si *objetSVG* n'est pas un objet texte SVG, une erreur est générée.

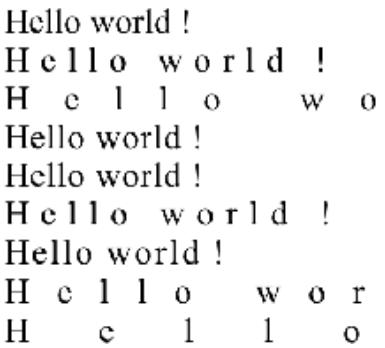
Le paramètre optionnel *unité* permet de préciser l'unité de la valeur d'espacement. La valeur par défaut est "%".

Si *espacement* vaut -1, la valeur d'espacement est fixée sur 'normal'.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/text.html#LetterSpacingProperty

Exemple

Exemples de variations d'espacement :




```
 //Référence
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;40;"";36)

$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;120;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"em")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"px")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"pt")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"pc")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;280;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"mm")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"cm")
```

```
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;360;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"in")
```

SVG_SET_TEXT_RENDERING

SVG_SET_TEXT_RENDERING (objetSVG ; rendu)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'un élément texte
rendu	Texte	⇒	Valeur de rendu

Description

La commande *SVG_SET_TEXT_RENDERING* permet de définir les compromis à utiliser pour le rendu du texte de l'objet texte désigné par *objetSVG*. Si *objetSVG* n'est pas un objet texte SVG, une erreur est générée.

Le paramètre *rendu* peut prendre l'une des valeurs suivantes : "auto", "optimizeSpeed", "optimizeLegibility", "geometricPrecision" ou "inherit". Dans le cas contraire, une erreur est générée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#TextRenderingProperty

SVG_SET_TEXT_WRITING_MODE

SVG_SET_TEXT_WRITING_MODE (objetSVG ; modeEcriture)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d'un élément texte
modeEcriture	Texte	➡	Sens de l'écriture

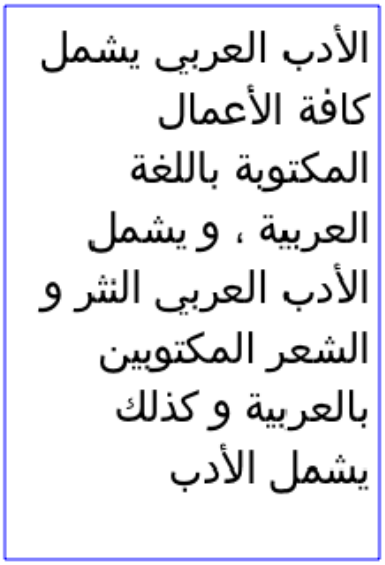
Description

La commande *SVG_SET_TEXT_WRITING_MODE* permet de définir si la direction d'écriture de l'objet texte désigné par *objetSVG* sera de gauche à droite, de droite à gauche ou de bas en haut. Si *objetSVG* n'est pas un objet texte SVG, une erreur est générée.

Le paramètre *modeEcriture* peut prendre une des valeurs suivantes : "lr-tb", "rl-tb", "tb-rl", "lr", "rl", "tb" ou "inherit ". Dans le cas contraire, une erreur est générée.

Exemple

Ecriture de droite à gauche :




```
 //Cadre
SVG_New_rect($Dom_SVG;5;5;210;310;0;0;"blue";"none")

 //Texte
$Dom_text:=SVG_New_textArea($Dom_SVG;$Txt_sample;10;10;200;300;"Baghdad 'Arial Unicode MS'";25)
SVG_SET_TEXT_WRITING_MODE($Dom_text;"rl")
```

SVG_SET_TEXTAREA_TEXT

SVG_SET_TEXTAREA_TEXT (objetSVG ; leTexte)

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d’un élément texte
leTexte	Texte	⇒	Texte à définir

Description

La commande **SVG_SET_TEXTAREA_TEXT** permet de fixer/remplacer le contenu textuel de l’objet texte désigné par *objetSVG*. Si *objetSVG* n’est pas un objet "textArea", une erreur est générée.

Les caractères retour à la ligne sont automatiquement remplacés par des éléments "<tbreak/>".

Utilitaires

- SVG_ABOUT
- SVG_Count_elements
- SVG_ELEMENTS_TO_ARRAYS
- SVG_Estimate_weight
- SVG_Find_ID
- SVG_Get_options
- SVG_Get_root_reference
- SVG_Get_version
- SVG_Is_reference_valid
- SVG_Post_comment
- SVG_Read_element_type
- SVG_Read_last_error
- SVG_References_array
- SVG_ROTATION_CENTERED
- SVG_SCALING_CENTERED
- SVG_Set_error_handler
- SVG_SET_OPTIONS
- SVGTool_SET_VIEWER_CALLBACK
- SVGTool_SHOW_IN_VIEWER

SVG_ABOUT

SVG_ABOUT
Ne requiert pas de paramètre

Description

La commande *SVG_ABOUT* affiche un dialogue avec le logo 4D SVG et indiquant le numéro de version du composant :

SVG_Count_elements

SVG_Count_elements (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence SVG
Résultat	Entier long	⇐	Nombre d'objets

Description

La commande *SVG_Count_elements* retourne le nombre d'objets graphiques contenus dans l'*objetSVG* passé en paramètre. Les groupes comptent pour un objet. Pour connaître le nombre d'objets graphiques d'un groupe, passez sa référence à la commande. Si la référence SVG n’est pas valide, une erreur est générée.

SVG_ELEMENTS_TO_ARRAYS

SVG_ELEMENTS_TO_ARRAYS (objetSVG ; pointeurTabRef {; pointTabTypes {; pointTabNoms}})

Paramètre	Type		Description
objetSVG	Alpha	➡	Référence SVG
pointeurTabRef	Pointeur	➡	Tableau chaîne des références d'objets
pointTabTypes	Pointeur	➡	Tableau chaîne des types d'objets
pointTabNoms	Pointeur	➡	Tableau chaîne des id d'objets

Description

La commande *SVG_ELEMENTS_TO_ARRAYS* remplit le tableau pointé par *pointTabRefs* avec les références des objets graphiques de premier niveau pour la référence SVG passée dans *objetSVG*.

Si le pointeur optionnel *pointTabTypes* est passé, le tableau sera renseigné avec le type des objets.

Si le pointeur optionnel *pointTabNoms* est passé, le tableau sera renseigné avec l'id des objets.

Les groupes comptent pour un objet. Pour retourner ces informations pour les objets graphiques d'un groupe, passez sa référence à la commande.

Si *objetSVG* n’est pas valide ou si cet attribut n'existe pas, une erreur est générée.

SVG_Estimate_weight

SVG_Estimate_weight (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'un document SVG
Résultat	Réel	⇐	Poids en octets du document SVG

Description

La commande *SVG_Estimate_weight* retourne la taille en octets de l'arbre SVG dont la référence est passée dans le paramètre *objetSVG*. Si *objetSVG* n’est pas une référence valide, une erreur est générée.

SVG_Find_ID

SVG_Find_ID (objetSVG ; nom) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'objet SVG
nom	Chaîne	→	id de l'élément SVG
Résultat	Ref_SVG	↩	Référence de l'élément

Description

La commande *SVG_Find_ID* retourne la référence de l'élément dont l'ID est passé dans le paramètre *nom*, appartenant à la structure SVG dont l'élément racine est passé dans le paramètre *objetSVG*. Si l'élément n'est pas trouvé, une erreur est générée.

SVG_Get_options

SVG_Get_options -> Résultat

Paramètre	Type		Description
Résultat	Entier long		Options

Description

La commande *SVG_Get_options* retourne un entier long représentant un tableau de 32 bits, chaque bit pouvant décrire une option du composant. Vous pouvez utiliser les **Opérateurs sur les bits** de 4D pour tester (??) l'état d'une option, l'activer (?+) ou la désactiver (?-).
Les options actuellement disponibles sont :

Bit	Option	Défaut
1	Attribuer un ID automatiquement lors de la création d'un élément	0 (inactivé)
2	Fermer automatiquement les objets qui peuvent l'être	0 (inactivé)
3	Créer les objets avec un fond	1 (activé)
4	Coordonnées absolues pour les tracés	1 (activé)
5	Créer un code plus lisible	0 (inactivé)
6	Émettre un bip lorsqu'une erreur survient	1 (activé)
7	Ne pas afficher les erreurs 4D	0 (inactivé)
8	Images transparentes	1 (activé)
9	Utiliser origine trigonométrique	0 (inactivé)
10	Substitution 'Arial' automatique	1 (activé)
11	Utiliser le rendu 'crispEdges' par défaut pour un nouveau dessin	0 (inactivé)
12	Contrôle des paramètres	1 (activé)
13	Conserver les espaces superflus	0 (inactivé)
14	Rotation centrée	0 (inactivé)

- *Attribuer un ID automatiquement lors de la création d'un élément*
  Si cette option est activée, lorsque le composant crée un nouvel élément, il ajoute et renseigne systématiquement un attribut 'id' pour l'objet créé s'il n'est pas précisé.

- *Fermer automatiquement les objets*
  Si cette option est activée, les objets créés avec les commandes *SVG_New_arc* et *SVG_New_polyline_by_arrays* seront fermés automatiquement.

- *Créer les objets avec un fond*
  Si cette option est activée, les objets fermés seront créés avec une couleur de fond, sinon le fond est transparent.

- *Coordonnées absolues pour les tracés*
  Lors du dessin de tracés avec les commandes *SVG_PATH_MOVE_TO*, *SVG_PATH_LINE_TO*, *SVG_PATH_CURVE* et *SVG_PATH_ARC*, les coordonnées passées seront interprétées comme absolues si cette option est activée ou comme relatives sinon.

- *Créer un code plus lisible*
  Cette option permet de créer un code indenté et aéré mais par conséquent plus lourd, son activation est surtout intéressante durant la phase de débogage.

- *Émettre un bip lorsqu'une erreur survient*
  Lorsqu'une erreur se produit et qu'aucune méthode d'erreur de la base hôte n'a été installée avec la commande *SVG_Set_error_handler*, un bip est émis si cette option est activée.

- *Ne pas afficher les erreurs 4D*
  Cette option activée par défaut bloque l'affichage des erreurs 4D en installant une méthode de gestion d'erreur propre au composant 4D SVG. Vous pouvez préférer ne pas utiliser cette gestion interne et laisser 4D afficher ces messages. Cela peut être utile en cours de débogage par exemple.

- *Images transparentes*
  Par défaut, les images SVG créées avec la commande *SVG_New* sont transparentes. Si vous désactivez cette option, les images seront sur fond blanc.

- *Utiliser origine trigonométrique*
  Par défaut, SVG met l'origine de l'échelle des degrés en haut (minuit). Cette option permet de passer les coordonnées en fonction du repère trigonométrique (3h ou 15mn) qui est souvent utilisé. La conversion est effectuée à la volée.

- *Substitution 'Arial' automatique*
  Par défaut, 4D SVG remplace la police 'Arial' par 'Arial Unicode MS','Arial', ce qui assure une meilleure compatibilité avec les caractères non romans (japonais par exemple). Dans certains cas, vous pouvez vouloir désactiver ce fonctionnement. Cette option permet de ne pas remplacer les polices 'Arial'.

- *Utiliser le rendu 'crispEdges' par défaut pour un nouveau dessin*
  L'attribut crispEdges (cf. **SVG_SET_SHAPE_RENDERING**) peut être forcé par défaut grâce à cette option.

- *Contrôle des paramètres*
  Par défaut, 4D SVG contrôle la validité des paramètres passés aux commandes. Une fois la phase de développement terminée, vous pouvez inactiver cette option afin d'accélérer sensiblement l'exécution du code.

- *Conserver les espaces superflus* (nouveauté 4D v14)
  Permet d'afficher plusieurs espaces adjacents dans les objets texte

- *Rotation centrée* (nouveauté 4D v14)
  Lorsque cette option est activée, la commande **SVG_SET_TRANSFORM_ROTATE** tentera d'effectuer une rotation centrée si ses 3e et 4e paramètres sont omis. Le centre de la rotation sera calculé à partir des attributs x, y, *width* et *height* de l'objet référencé. Si l'objet ne dispose pas de ces attributs, la rotation sera effectuée autour du point (0,0).

Exemple

Reportez-vous à l'exemple de la commande *SVG_SET_OPTIONS*.

SVG_Get_root_reference

SVG_Get_root_reference (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence d’objet SVG
Résultat	Ref_SVG	➲	Référence de l’élément racine

Description

La commande **SVG_Get_root_reference** retourne la référence de l’élément racine de l’objet SVG dont vous avez passé la référence dans *objetSVG*.

SVG_Get_version

SVG_Get_version -> Résultat

Paramètre	Type		Description
Résultat	Chaîne		Numéro de version

Description

La commande *SVG_Get_version* retourne sous forme alphanumérique le numéro de version du composant. La chaîne retournée comporte toujours le numéro de version et de sous-version (par exemple "11.0" pour la version 11 ou "11.3" pour la 3e mise à jour de la version 11). Lorsqu'il s'agit d'une version beta, le numéro de distribution est précisé, préfixé de la lettre "B" (par exemple "11.3B1" pour la beta 1 de la version 11.3).

SVG_Is_reference_valid

SVG_Is_reference_valid (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément SVG
Résultat	Booléen	↩	Vrai si la référence appartient à un élément SVG

Description

La commande *SVG_Is_reference_valid* retourne **Vrai** si la référence passée dans le paramètre *objetSVG* est celle d'un élément d'un arbre SVG. Si l'élément n'appartient pas à un arbre SVG, la commande retourne **Faux**. Si *objetSVG* n’est pas une référence valide, une erreur est générée.

SVG_Post_comment

SVG_Post_comment (objetSVG ; commentaire) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'objet SVG
commentaire	Texte	⇒	Texte à ajouter en tant que commentaire
Résultat	Ref_SVG	⇐	Référence du commentaire

Description

La méthode **SVG_Post_comment** ajoute le *commentaire* dans l'objet SVG désigné par le paramètre *objetSVG* sous forme de commentaire XML.

La méthode retourne la référence SVG du commentaire.

Exemple

Le code suivant :

```
C_TEXTE($comment)
$comment:="Modifié le "+Chaine(Date du jour)
$ref:=SVG_Post_comment($svg;$comment )
```

... ajoute dans l'objet SVG $svg :

```
<!--Modifié le 12/02/2012-->
```

SVG_Read_element_type

SVG_Read_element_type (objetSVG) -> Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	⇒	Référence d'élément SVG
Résultat		⇐	Type de l'élément

Description

La commande *SVG_Read_element_type* retourne le type de l'élément dont la référence est passée dans le paramètre *objetSVG*.

Si *objetSVG* n’est pas une référence valide ou si cet attribut n'existe pas, une erreur est générée.

SVG_Read_last_error

SVG_Read_last_error -> Résultat

Paramètre	Type		Description
Résultat	Entier long		Numéro de la dernière erreur

Description

La commande *SVG_Read_last_error* retourne le numéro de la dernière erreur survenue lors de l'exécution d'une commande du composant 4D SVG et réinitialise cette erreur.

Le numéro d'erreur retourné peut être spécifique à une commande du composant ou une erreur générée par 4D. Les erreurs générées par le composant sont :

8850	Nombre de paramètres insuffisant
8851	Type de paramètre non valide
8852	Référence non valide
8853	Valeur incorrecte pour un attribut
8854	L'élément n'accepte pas cette commande
8855	Nom d'objet (symbole, marqueur, filtre...) invalide (ID non trouvé dans le document)
8856	Le fichier DTD n'a pas été trouvé.
8857	Valeur incorrecte pour un paramètre
8858	Erreur inconnue

Exemple 1

Soit la méthode "gest_SVG_error" décrite dans l'exemple de la commande *SVG_Set_error_handler* :

```
  `Installation de la méthode de gestion d'erreur
$Error_Method_Txt:=SVG_Set_error_handler("gest_SVG_error")
  `Désormais c'est la méthode gest_SVG_error qui sera exécutée en cas d'erreur

  `Création d'un nouveau document SVG
$SVG:=SVG_New(1200;900;"Test Composant SVG Component";"";Vrai)
SVG_SET_VIEWBOX($SVG;0;0;1500;1000)

Si(SVG_Read_last_error=0)

   ...

Sinon
  `La méthode gest_SVG_error a été appelée et a reçu le numéro d'erreur
Fin de si

  `Désinstallation de la méthode de gestion d'erreurs
SVG_Set_error_handler
```

Exemple 2

Soit la méthode ***gest_SVG_error*** suivante :

```
C_ENTIER LONG($1)
C_TEXTE($2)

  `Garder l'erreur et le contexte
NumErreur:=$1
NomCommande:=$2

  `Mettre la variable système OK à 0
OK:=0
```

Cette méthode peut être utilisée de la manière suivante :

```
  ` Installation de la méthode de gestion d'erreur
$Error_Method_Txt:=SVG_Set_error_handler("gest_SVG_error")

  ` Création d'un nouveau document SVG
$SVG:=SVG_New(1200;900;"Test composant SVG";"";Vrai)
SVG_SET_VIEWBOX($SVG;0;0;1500;1000)
Si(OK=1)
   ...
Sinon
   ALERTE("Erreur N°."+Chaine(NumErreur)+" pendant l'exécution de la commande
\""+NomCommande+"\"")
Fin de si

  ` Désinstallation de la méthode de gestion d'erreurs
SVG_Set_error_handler
```

SVG_References_array

SVG_References_array (pointeurTabRef) -> Résultat

Paramètre	Type		Description
pointeurTabRef	Pointeur	⇒	Tableau chaîne des références de documents
Résultat	Entier long	⇐	Nombre de références

Description

La commande *SVG_References_array* retourne dans le tableau pointé par *pointeurTabRef* la liste des références de documents SVG courantes. En résultat, la commande retourne le nombre de références trouvées.

SVG_References_array est utile en cours de débogage. A chaque fois qu'un document SVG est créé avec les commandes *SVG_New*, *SVG_Copy* ou *SVG_Open_file*, le composant ajoute la référence retournée par la commande dans un tableau interne. Quand le document SVG est libéré avec la commande *SVG_CLEAR*, le composant supprime la référence du tableau.

SVG_ROTATION_CENTERED

SVG_ROTATION_CENTERED (objetSVG ; angle)

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence objet SVG
angle	Réel	➡	Angle de rotation

Description

La commande **SVG_ROTATION_CENTERED** opère une rotation centrée sur l'objet SVG passé en référence dans le paramètre *objetSVG*. Ce type de rotation peut être appliqué seulement aux objets avec attributs "x", "y", "width" et "height".

Dans le paramètre *angle*, vous passez l'angle de rotation à fixer.

SVG_SCALING_CENTERED

SVG_SCALING_CENTERED (objetSVG ; échelle {; x ; y})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence SVG
échelle	Réel	➡	Echelle
x	Réel	➡	Axe des x
y	Réel	➡	Axe des y

Description

La commande **SVG_SCALING_CENTERED** opère une mise à l'échelle de l'image SVG passée en paramètre dans *objetSVG*.

Dans le paramètre *échelle*, vous passez une valeur positive (>1). Si vous passez 1, l'échelle est à 100%.

Dans les paramètres optionnels *x* et *y*, vous pouvez passer respectivement les coordonnées des axes x- et y- à partir du point central. Si vous ne passez pas ces paramètres, le point central est déterminé par les attributs de l'objet "x", "y", "width" et "height" (s'ils existent). Si ce type de transformation est appliqué sur un objet qui n'a pas ces attributs et que les paramètres optionnels *x* et *y* sont omis, une chaîne vide est retournée.

SVG_Set_error_handler

SVG_Set_error_handler {(méthode)} -> Résultat

Paramètre	Type		Description
méthode	Chaîne	➡	Nom de la méthode de la base hôte à installer
Résultat	Chaîne	⮌	Nom de la méthode précédemment installée

Description

La commande *SVG_Set_error_handler* permet d'installer la *méthode* de la base hôte en tant que méthode appelée en cas d'erreur et retourne le nom de la précédente méthode d'appel sur erreur.

Les commandes du composant 4D SVG effectuent un ensemble de vérifications élémentaires lorsqu'elles sont appelées : nombre de paramètres minimum, validité des références, de l'élément sur lequel une commande est appliquée... Le composant gère donc les erreurs de façon structurée et permet à la base hôte de récupérer les éventuelles erreurs.

En l'absence de modification du fonctionnement par défaut, si une erreur survient, un bip est émis et la commande est interrompue.

La base hôte peut récupérer dans une de ses méthodes le numéro de l'erreur et le nom de la commande à l'origine de l'interruption. Il suffit pour cela de l'installer via la commande *SVG_Set_error_handler*. Cette méthode recevra en premier paramètre le numéro de l'erreur et en second paramètre le nom de la commande. Elle sera appelée lorsqu'une erreur se produira et dans ce cas aucun bip ne sera généré par le composant.

Si méthode est omis ou si vous passez une chaîne vide dans ce paramètre, la méthode est désinstallée et le comportement par défaut est réactivé.

Note : La méthode de la base hôte qui sera appelée par le composant 4D SVG doit avoir la propriété "Partagée entre composants et base hôte".

Exemple

Installation de la méthode ***gest_SVG_error*** (méthode de la base hôte) comme méthode d'appel sur erreur :

```
$erreur:=SVG_Set_error_handler("gest_SVG_error")
```

Code de la méthode :

```
` Méthode gest_SVG_error
ALERTE("Erreur No."+Chaine($1)+" pendant l'exécution de la commande \""+$2+"\"")
```

SVG_SET_OPTIONS

SVG_SET_OPTIONS {(options)}

Paramètre	Type		Description
options	Entier long	➡	Options du composant 4D SVG

Description

La commande *SVG_SET_OPTIONS* permet de fixer les options du composant 4D SVG avec l'entier long *options*. Pour plus d'informations sur le contenu de *options*, reportez-vous à la description de la commande *SVG_Get_options*.

Comme toutes les options sont fixées simultanément, cette commande doit être précédée d'un appel à la commande *SVG_Get_options* puis de l'utilisation des **Opérateurs sur les bits** de 4D.

Si le paramètre *option* est omis, toutes les options sont réinitialisées à leur valeur par défaut (cf. commande *SVG_Get_options*).

Exemple 1

Créer un code plus lisible :

```
$Options :=SVG_Get_options
$Options :=$Options ?+5  `activer l'option
SVG_SET_OPTIONS($Options)
```

Exemple 2

Dessiner un graphique en camembert :

```
$svg:=SVG_New

 `Activer la fermeture automatique des objets
SVG_SET_OPTIONS(SVG_Get_options?+2)

SVG_New_arc($svg;100;100;90;0;105;"gray";"lightcoral";1)
SVG_New_arc($svg;100;100;90;105;138;"gray";"lightskyblue";1)
SVG_New_arc($svg;100;100;90;138;230;"gray";"lightgreen";1)
SVG_New_arc($svg;100;100;90;230;270;"gray";"lightsteelblue";1)
SVG_New_arc($svg;100;100;90;270;360;"gray";"lightyellow";1)
```

Exemple 3

Afficher plusieurs espaces dans des objets texte en utilisant l'option (13) *Keep extra spaces* (ajoutée en v14) :

```
$Txt_buffer:="abc        def"
$Dom_text:=SVG_New_textArea($Dom_svg;$Txt_buffer;50;50)
```

est affiché "abc def"

```
SVG_SET_OPTIONS(SVG_Get_options?+13)  // pour conserver les espaces dans l'objet texte
$Txt_buffer:="abc        def"
$Dom_text:=SVG_New_textArea($Dom_svg;$Txt_buffer;50;50)
```

est affiché "abc def"

SVGTool_SET_VIEWER_CALLBACK

SVGTool_SET_VIEWER_CALLBACK (nomMéthode)

Paramètre	Type		Description
nomMéthode	Texte	➡	Nom de méthode projet 4D

Description

La commande *SVGTool_SET_VIEWER_CALLBACK* permet d'installer *nomMéthode* comme méthode projet appelée lors des événements Sur clic et Sur survol survenant sur l'image affichée par la commande *SVGTool_SHOW_IN_VIEWER*.

Cette méthode reçoit un paramètre texte qui est l'id de l'élément survolé ou cliqué tel que fourni par la commande 4D **SVG Chercher ID element par coordonnees**.

Vous devez impérativement déclarer ce paramètre dans la méthode projet *nomMéthode* de la base hôte en y insérant la ligne C_TEXTE($1).

Vous devez affecter la propriété "Partagée entre composants et base hôte" à la méthode *nomMéthode*.

Si la méthode n'existe pas ou n'est pas partagée, l'erreur -10508 est générée par 4D.

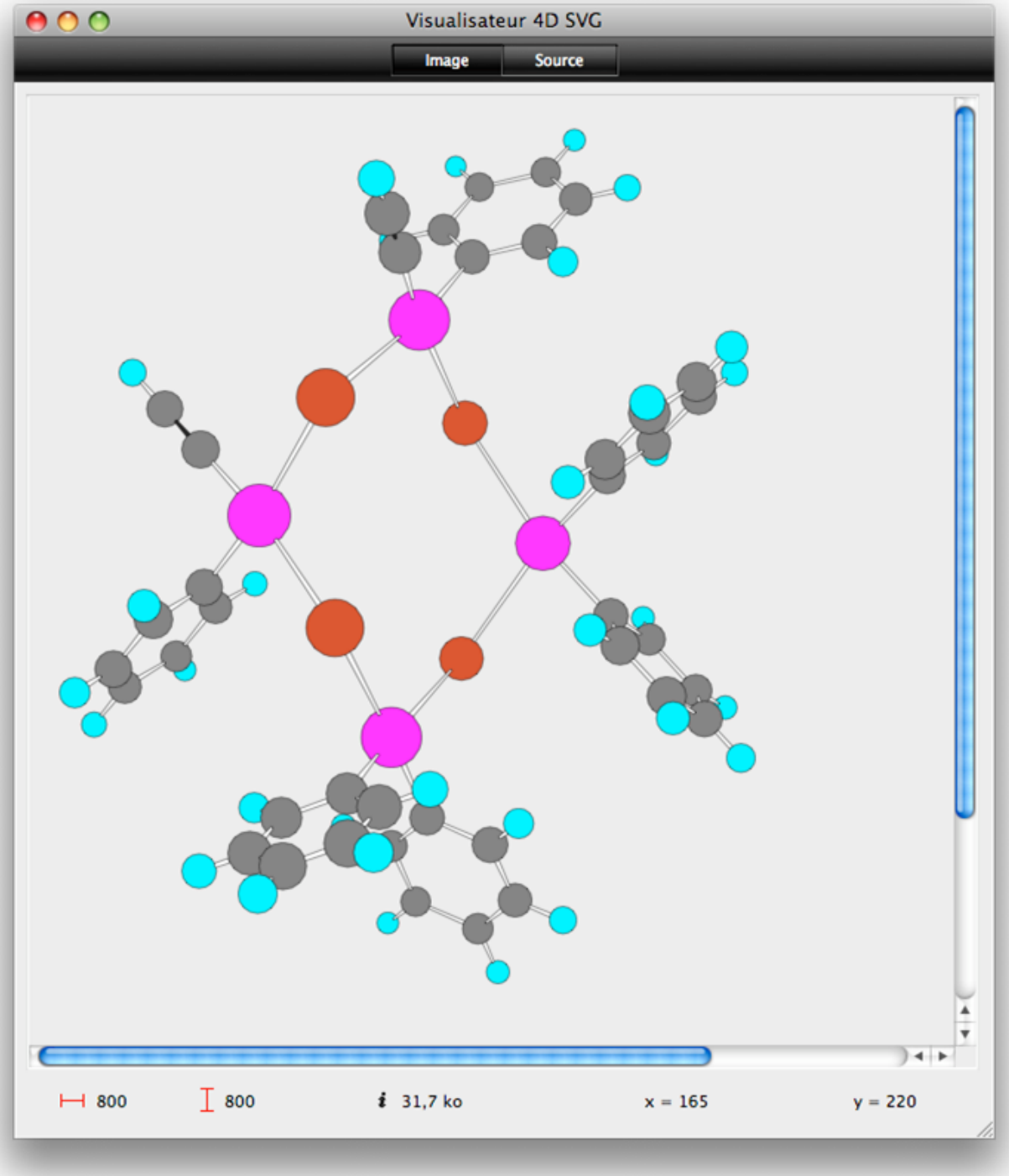
SVGTool_SHOW_IN_VIEWER

SVGTool_SHOW_IN_VIEWER (objetSVG {; sources})

Paramètre	Type		Description
objetSVG	Ref_SVG	➡	Référence de l'image à afficher
sources	Chaîne	➡	Ouvre directement le visualisateur SVG dans la page source

Description

La commande *SVGTool_SHOW_IN_VIEWER* affiche l'image SVG définie par *objetSVG* dans une fenêtre du Visualisateur SVG. Cet outil est fourni avec le composant SVG :



Si vous passez le paramètre optionnel *sources* (ajouté en v14), le visualisateur est directement ouvert dans la page source.

Pour plus d'informations sur le Visualisateur SVG, reportez-vous à la section **Outils de développement**.

Constantes 4D SVG

SVG Colors (Names)
SVG Colors (RGB)

SVG Colors (Names)

Constante	Type	Valeur	Comment
color SVG aliceblue	Chaîne	aliceblue	
color SVG antiquewhite	Chaîne	antiquewhite	
color SVG aqua	Chaîne	aqua	
color SVG aquamarine	Chaîne	aquamarine	
color SVG azure	Chaîne	azure	
color SVG beige	Chaîne	beige	
color SVG bisque	Chaîne	bisque	
color SVG black	Chaîne	black	
color SVG blanchedalmond	Chaîne	blanchedalmond	
color SVG blue	Chaîne	blue	
color SVG blueviolet	Chaîne	blueviolet	
color SVG brown	Chaîne	brown	
color SVG burlywood	Chaîne	burlywood	
color SVG cadetblue	Chaîne	cadetblue	
color SVG chartreuse	Chaîne	chartreuse	
color SVG chocolate	Chaîne	chocolate	
color SVG coral	Chaîne	coral	
color SVG cornflowerblue	Chaîne	cornflowerblue	
color SVG cornsilk	Chaîne	cornsilk	
color SVG crimson	Chaîne	crimson	
color SVG cyan	Chaîne	cyan	
color SVG darkblue	Chaîne	darkblue	

Constante	Type	Valeur	Comment
color SVG darkcyan	Chaîne	darkcyan	
color SVG darkgoldenrod	Chaîne	darkgoldenrod	
color SVG darkgray	Chaîne	darkgray	
color SVG darkgreen	Chaîne	darkgreen	
color SVG darkgrey	Chaîne	darkgrey	
color SVG darkkhaki	Chaîne	darkkhaki	
color SVG darkmagenta	Chaîne	darkmagenta	
color SVG darkolivegreen	Chaîne	darkolivegreen	
color SVG darkorange	Chaîne	darkorange	
color SVG darkorchid	Chaîne	darkorchid	
color SVG darkred	Chaîne	darkred	
color SVG darksalmon	Chaîne	darksalmon	
color SVG darkseagreen	Chaîne	darkseagreen	
color SVG darkslateblue	Chaîne	darkslateblue	
color SVG darkslategray	Chaîne	darkslategray	
color SVG darkslategrey	Chaîne	darkslategrey	
color SVG darkturquoise	Chaîne	darkturquoise	
color SVG darkviolet	Chaîne	darkviolet	
color SVG deeppink	Chaîne	deeppink	
color SVG deepskyblue	Chaîne	deepskyblue	
color SVG dimgray	Chaîne	dimgray	
color SVG dimgrey	Chaîne	dimgrey	

Constante	Type	Valeur	Comment
color SVG dodgerblue	Chaîne	dodgerblue	
color SVG firebrick	Chaîne	firebrick	
color SVG floralwhite	Chaîne	floralwhite	
color SVG forestgreen	Chaîne	forestgreen	
color SVG fuchsia	Chaîne	fuchsia	
color SVG gainsboro	Chaîne	gainsboro	
color SVG ghostwhite	Chaîne	ghostwhite	
color SVG gold	Chaîne	gold	
color SVG goldenrod	Chaîne	goldenrod	
color SVG gray	Chaîne	gray	
color SVG green	Chaîne	green	
color SVG greenyellow	Chaîne	greenyellow	
color SVG grey	Chaîne	grey	
color SVG honeydew	Chaîne	honeydew	
color SVG hotpink	Chaîne	hotpink	
color SVG indianred	Chaîne	indianred	
color SVG indigo	Chaîne	indigo	
color SVG ivory	Chaîne	ivory	
color SVG khaki	Chaîne	khaki	
color SVG lavender	Chaîne	lavender	
color SVG lavenderblush	Chaîne	lavenderblush	
color SVG lawngreen	Chaîne	lawngreen	

Constante	Type	Valeur	Comment
color SVG lemonchiffon	Chaîne	lemonchiffon	
color SVG lightblue	Chaîne	lightblue	
color SVG lightcoral	Chaîne	lightcoral	
color SVG lightcyan	Chaîne	lightcyan	
color SVG lightgoldenrodyellow	Chaîne	lightgoldenrodyellow	
color SVG lightgray	Chaîne	lightgray	
color SVG lightgreen	Chaîne	lightgreen	
color SVG lightgrey	Chaîne	lightgrey	
color SVG lightpink	Chaîne	lightpink	
color SVG lightsalmon	Chaîne	lightsalmon	
color SVG lightseagreen	Chaîne	lightseagreen	
color SVG lightskyblue	Chaîne	lightskyblue	
color SVG lightslategray	Chaîne	lightslategray	
color SVG lightslategrey	Chaîne	lightslategrey	
color SVG lightsteelblue	Chaîne	lightsteelblue	
color SVG lightyellow	Chaîne	lightyellow	
color SVG lime	Chaîne	lime	
color SVG limegreen	Chaîne	limegreen	
color SVG linen	Chaîne	linen	
color SVG magenta	Chaîne	magenta	
color SVG maroon	Chaîne	maroon	
color SVG mediumaquamarine	Chaîne	mediumaquamarine	

Constante	Type	Valeur	Comment
color SVG mediumblue	Chaîne	mediumblue	
color SVG mediumorchid	Chaîne	mediumorchid	
color SVG mediumpurple	Chaîne	mediumpurple	
color SVG mediumseagreen	Chaîne	mediumseagreen	
color SVG mediumslateblue	Chaîne	mediumslateblue	
color SVG mediumspringgreen	Chaîne	mediumspringgreen	
color SVG mediumturquoise	Chaîne	mediumturquoise	
color SVG mediumvioletred	Chaîne	mediumvioletred	
color SVG midnightblue	Chaîne	midnightblue	
color SVG mintcream	Chaîne	mintcream	
color SVG mistyrose	Chaîne	mistyrose	
color SVG moccasin	Chaîne	moccasin	
color SVG navajowhite	Chaîne	navajowhite	
color SVG navy	Chaîne	navy	
color SVG oldlace	Chaîne	oldlace	
color SVG olive	Chaîne	olive	
color SVG olivedrab	Chaîne	olivedrab	
color SVG orange	Chaîne	orange	
color SVG orangered	Chaîne	orangered	
color SVG orchid	Chaîne	orchid	
color SVG palegoldenrod	Chaîne	palegoldenrod	
color SVG palegreen	Chaîne	palegreen	

Constante	Type	Valeur	Comment
color SVG paleturquoise	Chaîne	paleturquoise	
color SVG palevioletred	Chaîne	palevioletred	
color SVG papyawhip	Chaîne	papyawhip	
color SVG peachpuff	Chaîne	peachpuff	
color SVG peru	Chaîne	peru	
color SVG pink	Chaîne	pink	
color SVG plum	Chaîne	plum	
color SVG powderblue	Chaîne	powderblue	
color SVG purple	Chaîne	purple	
color SVG red	Chaîne	red	
color SVG rosybrown	Chaîne	rosybrown	
color SVG royalblue	Chaîne	royalblue	
color SVG saddlebrown	Chaîne	saddlebrown	
color SVG salmon	Chaîne	salmon	
color SVG sandybrown	Chaîne	sandybrown	
color SVG seagreen	Chaîne	seagreen	
color SVG seashell	Chaîne	seashell	
color SVG sienna	Chaîne	sienna	
color SVG silver	Chaîne	silver	
color SVG skyblue	Chaîne	skyblue	
color SVG slateblue	Chaîne	slateblue	
color SVG slategray	Chaîne	slategray	

Constante	Type	Valeur	Comment
color SVG slategrey	Chaîne	slategrey	
color SVG snow	Chaîne	snow	
color SVG springgreen	Chaîne	springgreen	
color SVG steelblue	Chaîne	steelblue	
color SVG tan	Chaîne	tan	
color SVG teal	Chaîne	teal	
color SVG thistle	Chaîne	thistle	
color SVG tomato	Chaîne	tomato	
color SVG turquoise	Chaîne	turquoise	
color SVG violet	Chaîne	violet	
color SVG wheat	Chaîne	wheat	
color SVG white	Chaîne	white	
color SVG whitesmoke	Chaîne	whitesmoke	
color SVG yellow	Chaîne	yellow	
color SVG yellowgreen	Chaîne	yellowgreen	

SVG Colors (RGB)

Constante	Type	Valeur	Comment
color RGB aliceblue	Entier long	15792383	
color RGB antiquewhite	Entier long	16444375	
color RGB aqua	Entier long	65535	
color RGB aquamarine	Entier long	8388564	
color RGB azure	Entier long	15794175	
color RGB beige	Entier long	16119260	
color RGB bisque	Entier long	16770244	
color RGB black	Entier long	0	
color RGB blanchedalmond	Entier long	16772045	
color RGB blue	Entier long	255	
color RGB blueviolet	Entier long	9055202	
color RGB brown	Entier long	10824234	
color RGB burlywood	Entier long	14596231	
color RGB cadetblue	Entier long	6266528	
color RGB chartreuse	Entier long	8388352	
color RGB chocolate	Entier long	13789470	
color RGB coral	Entier long	16744272	
color RGB cornflowerblue	Entier long	6591981	
color RGB cornsilk	Entier long	16775388	
color RGB crimson	Entier long	14423100	
color RGB cyan	Entier long	65535	
color RGB darkblue	Entier long	139	

Constante	Type	Valeur	Comment
color RGB darkcyan	Entier long	35723	
color RGB darkgoldenrod	Entier long	12092939	
color RGB darkgray	Entier long	11119017	
color RGB darkgreen	Entier long	25600	
color RGB darkgrey	Entier long	11119017	
color RGB darkkhaki	Entier long	12433259	
color RGB darkmagenta	Entier long	9109643	
color RGB darkolivegreen	Entier long	5597999	
color RGB darkorange	Entier long	16747520	
color RGB darkorchid	Entier long	10040012	
color RGB darkred	Entier long	9109504	
color RGB darksalmon	Entier long	15308410	
color RGB darkseagreen	Entier long	9419919	
color RGB darkslateblue	Entier long	4734347	
color RGB darkslategray	Entier long	3100495	
color RGB darkslategrey	Entier long	3100495	
color RGB darkturquoise	Entier long	52945	
color RGB darkviolet	Entier long	9699539	
color RGB deeppink	Entier long	16716947	
color RGB deepskyblue	Entier long	49151	
color RGB dimgray	Entier long	6908265	
color RGB dodgerblue	Entier long	2003199	

Constante	Type	Valeur	Comment
color RGB firebrick	Entier long	11673122	
color RGB forestgreen	Entier long	2263842	
color RGB fuchsia	Entier long	16711935	
color RGB gainsboro	Entier long	14474460	
color RGB ghostwhite	Entier long	16316671	
color RGB gold	Entier long	16766720	
color RGB goldenrod	Entier long	14329120	
color RGB gray	Entier long	8421504	
color RGB green	Entier long	32768	
color RGB greenyellow	Entier long	11403055	
color RGB grey	Entier long	8421504	
color RGB honeydew	Entier long	15794160	
color RGB hotpink	Entier long	16738740	
color RGB indianred	Entier long	13458524	
color RGB indigo	Entier long	4915330	
color RGB ivory	Entier long	16777200	
color RGB khaki	Entier long	15787660	
color RGB lavender	Entier long	15132410	
color RGB lavenderblush	Entier long	16773365	
color RGB lawngreen	Entier long	8190976	
color RGB lemonchiffon	Entier long	16775885	
color RGB lightblue	Entier long	11393254	

Constante	Type	Valeur	Comment
color RGB lightcoral	Entier long	1576136	
color RGB lightcyan	Entier long	14745599	
color RGB lightgoldenrodyellow	Entier long	16448210	
color RGB lightgray	Entier long	13882323	
color RGB lightgreen	Entier long	9498256	
color RGB lightgrey	Entier long	13882323	
color RGB lightpink	Entier long	16758465	
color RGB lightsalmon	Entier long	16752762	
color RGB lightseagreen	Entier long	2142890	
color RGB lightskyblue	Entier long	8900346	
color RGB lightslategray	Entier long	7833753	
color RGB lightslategrey	Entier long	7833753	
color RGB lightsteelblue	Entier long	11584734	
color RGB lightyellow	Entier long	16777184	
color RGB lime	Entier long	65280	
color RGB limegreen	Entier long	3329330	
color RGB linen	Entier long	16445670	
color RGB magenta	Entier long	16711935	
color RGB maroon	Entier long	8388608	
color RGB mediumaquamarine	Entier long	6737322	
color RGB mediumblue	Entier long	205	
color RGB mediumorchid	Entier long	12211667	

Constante	Type	Valeur	Comment
color RGB mediumpurple	Entier long	9662683	
color RGB mediumseagreen	Entier long	3978097	
color RGB mediumslateblue	Entier long	8087790	
color RGB mediumspringgreen	Entier long	64154	
color RGB mediumturquoise	Entier long	4772300	
color RGB mediumvioletred	Entier long	13047173	
color RGB midnightblue	Entier long	1644912	
color RGB mintcream	Entier long	16121850	
color RGB mistyrose	Entier long	16770273	
color RGB moccasin	Entier long	16770273	
color RGB navajowhite	Entier long	16768685	
color RGB navy	Entier long	128	
color RGB oldlace	Entier long	16643558	
color RGB olive	Entier long	8421376	
color RGB olivedrab	Entier long	7048739	
color RGB orange	Entier long	16753920	
color RGB orangered	Entier long	16729344	
color RGB orchid	Entier long	14315734	
color RGB palegoldenrod	Entier long	15657130	
color RGB palegreen	Entier long	10025880	
color RGB paleturquoise	Entier long	11529966	
color RGB palevioletred	Entier long	14381203	

Constante	Type	Valeur	Comment
color RGB papyawhip	Entier long	16773077	
color RGB peachpuff	Entier long	16767673	
color RGB peru	Entier long	13468991	
color RGB pink	Entier long	16761035	
color RGB plum	Entier long	14524637	
color RGB powderblue	Entier long	11591910	
color RGB purple	Entier long	8388736	
color RGB red	Entier long	16711680	
color RGB rosybrown	Entier long	12357519	
color RGB royalblue	Entier long	4286945	
color RGB saddlebrown	Entier long	9127187	
color RGB salmon	Entier long	16416882	
color RGB sandybrown	Entier long	16032864	
color RGB seagreen	Entier long	3050327	
color RGB seashell	Entier long	16774638	
color RGB sienna	Entier long	10506797	
color RGB silver	Entier long	12632256	
color RGB skyblue	Entier long	8900331	
color RGB slateblue	Entier long	6970061	
color RGB slategray	Entier long	7372944	
color RGB slategrey	Entier long	7372944	
color RGB snow	Entier long	16775930	

Constante	Type	Valeur	Comment
color RGB springgreen	Entier long	65407	
color RGB steelblue	Entier long	4620980	
color RGB tan	Entier long	13808780	
color RGB teal	Entier long	32896	
color RGB thistle	Entier long	14204888	
color RGB tomato	Entier long	16737095	
color RGB turquoise	Entier long	4251856	
color RGB violet	Entier long	15631086	
color RGB wheat	Entier long	16113331	
color RGB white	Entier long	16777215	
color RGB whitesmoke	Entier long	16119285	
color RGB yellow	Entier long	16776960	
color RGB yellowgreen	Entier long	10145074	

Annexes

Annexe B, Liens externes

Annexe B, Liens externes

Voici une liste de liens utiles relatifs à la norme SVG :

Présentation
http://www.w3.org/Graphics/SVG/

Spécification
http://www.w3.org/TR/SVG12/ (Status : Working Draft 13 April 2005)
http://www.yoyodesign.org/doc/w3c/svg1/ (Traduction française (Version 1.0))

Tutoriaux
http://www.w3.org/Consortium/Offices/Presentations/SVG/1.svg

Communauté
http://svg.org/
http://www.openclipart.org/
http://www.gosvg.net/