AJUI Banner

Manuel d'utilisation

Version Control	Date	Commentaire (Changement)	Auteur
1.0	11.04.2019	Première version	Gary Criblez
1.1	02.05.2019	CornerRibbon et modification HDI	Maurice Inzirillo
1.5	17.06.2019	Spinner animation	Gary Criblez
1.5.1	11.06.2019	Compatibilité avec les sous-formulaires	Gabriel Inzirillo
1.6	10.09.2019	CornerRibbon – positionnement spécifique	Gary Criblez

1 Introduction

1.1 Qu'est-ce que AJUI Banner

AJUI_Banner est basé sur une idée originale proposée lors d'une formation faite par l'expert 4D Olivier Deschanels de 4D SAS, que nous avons implémeté sous forme de composant 4D.

AJUI Banner est un composant développé avec 4D 17 R3. Son utilisation est destinée aux développeurs 4D. Il permet de générer des bannières dans le contexte d'un formulaire.

Le but de ce composant est de permettre aux développeurs 4D d'enrichir leur interface de façon simple et rapide en automatisant la génération d'éléments visuelles permettant de fournir clairement et rapidement des informations aux utilisateurs.

Le composant permet de générer une instance d'objet bannière donnant accès aux différentes propriétés qui lui sont associées. Celles-ci peuvent être manipulées à l'aide de formule intégrées également à cette instance (nous reviendrons dessus sur le chapitre concernant les propriétés et le chapitre de prise en main). L'appel de la génération de l'image ce fait également à partir de l'objet. La création des images se fait à l'aide des méthodes SVG disponibles dans 4D.

L'image est ensuite associée à une variable image dans le formulaire cible et elle sera finalement correctement placée en fonction du type de bannière choisit.

1.2 Les bannières

Les bannières sont différenciées en deux formats ou types dénommés ruban d'angle « cornerRibbon » ou fenêtre « window ».

Une bannière est forcément associée à une variable formulaire de type image (voir chapitre prise en main). Celle-ci, va permettre d'accueillir la bannière. Le placement de la variable image sera également gérée par le composant en fonction du type choisit.

L'affichage de la variable image est à la charge du développeur.

1.2.1 Type « cornerRibbon »

La bannière ruban d'angle « cornerRibbon » est une image qui va être placée à l'un des quatre coins d'un formulaire.

L'image représente un bandeau qui va être disposé en diagonal en fonction de sa position. Son but principal est de fournir une information claire et distincte à l'utilisateur.

Attention : étant donné que c'est une image, si vous placez l'objet image qui va le contenir au premier plan et que vous avez des champs ou variables en dessous vous ne pourrez pas y accéder, même si ceux-ci sont visibles. Prenez soin, dans sa définition, de bien la dimensionner dans ses propriétés (taille et choix du coté) et de lui fournir une place n'occultant pas l'utilisation du formulaire. Même si visuellement elle représente un ruban d'angle, l'image qui le contient est de forme carrée.

Comment l'utiliser

L'utilisation du ruban d'angle « cornerRibbon » est utile lorsqu'on veut donner une information importante à l'utilisateur sur un jeu de données. Par exemple, si l'enregistrement chargé dans le formulaire est actuellement verouillé, vous pourrez informer via la bannière de l'état de celui-ci.

Sur le même principe, on peut imaginer de signaler lorsqu'on charge une facture, le fait qu'elle soit réglée ou qu'un client est considéré comme mauvais payeur ou encore afficher un niveau de priorité. D'une façon générale, l'idée est que la bannière attirera l'attention de l'utilisateur pour lui indiquer une information importante.

1.2.2 Type « window »

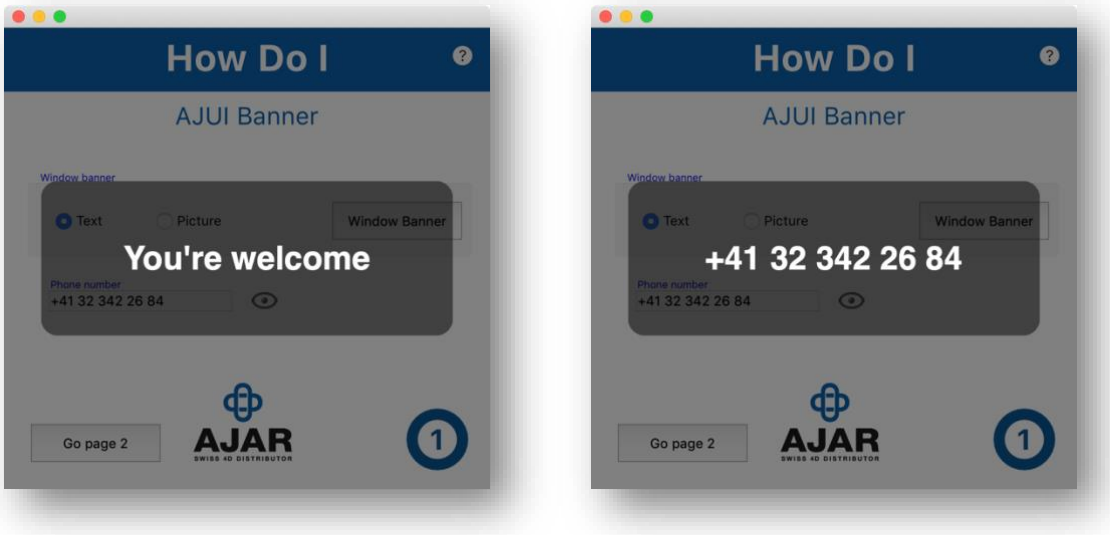
La bannière « window » est une image recouvrant l'ensemble du formulaire associé, elle doit donc être placée au premier plan de la page du formulaire. Elle peut contenir soit du texte ou une image.

Dans le cas du texte, AJUI_Banner dessine une première forme rectangulaire, servant à couvrir l'ensemble du formulaire, à laquelle est associé une couleur de fond avec un taux d'opacité. Puis, il centre une deuxième forme, la bannière (rectangle central) et son contenu textuel.

Dans le cas de l'utilisation d'une image, AJUI_Banner applique également un premier rectangle de recouvrement global et ensuite il centre l'image associée.

Le but de la bannière fenêtre « window » est principalement de signaler un traitement en cours à l'utilisateur.

Comment l'utiliser

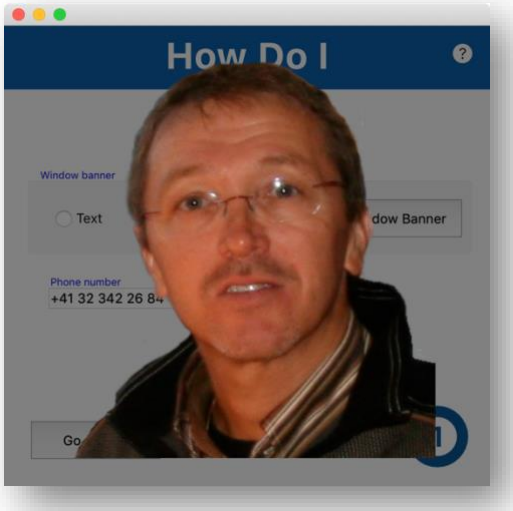


L'utilisation de la bannière fenêtre « window » est intéressante principalement pour deux situations. Si un traitement est lancé et qu'il peut prendre un certain temps, il est bon de le signaler à l'utilisateur et de bloquer toutes actions possible sur l'interface de sa part sur le formulaire courant. Il est aussi possible d'utiliser ce mode pour permettre l'affichage en gros plan d'une information utile momentanément. Par exemple afficher et de façon distinctive un numéro de téléphone, un numéro de série, etc.

Le deuxième usage possible, consiste à afficher une image. On peut ainsi imaginer afficher une image en overlay (fond noir semi-transparent) permettant à l'utilisateur de s'informer sur l'utilisation des champs, boutons, etc. dans un formulaire complexe à appréhender (figure de gauche). Ou afficher la photo d'un visage ou d'un objet, etc. (figure de droite).

Précision : Le type « window » va permettre de bloquer principalement les événements liés aux interactions avec la souris comme le clique, le focus, etc. étant donné qu'il se repercute sur l'image à l'avant-plan. Cependant des interactions avec le clavier comme la tabulation ne peuvent pas être

stoppées par la bannière. Faites donc attention à bien gérer les comportements que la bannière ne peut contrecarrer.

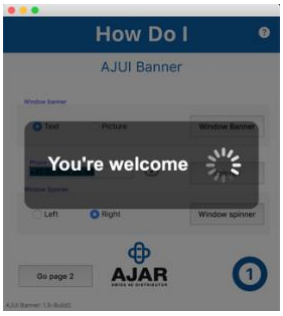


1.2.3 Types « spinner » et « windowSpinner »

Les bannières « spinners » sont des extensions de la bannière « window ». Le « spinner » est image représentant une sorte de roue qui est animé afin de suggérer à l'utilisateur un temps de chargement. Nous utilisons un Worker afin de réaliser l'effet d'animation.

Le premier type permet d'afficher le « spinner » seul au centre de l'écran et en y ajoutant un arrière-plan comme pour le type « window ».

Le second type « windowSpinner » va faire comme le précédent sauf qu'une zone de texte y est inclus est que le « spinner » sera situé à droite ou à gauche du texte.



1.2.4 Restrictions

Du fait que le langage SVG n'offre pas la possibilité d'un alignement vertical dans le cadre des zones de texte (TextArea) et par souci de cohérence de notre part, nous avons pris le parti de centrer verticalement le texte en fixant une contrainte :

- **Le texte doit tenir sur une seule et unique ligne !**

1.3 Compatibilité avec les sous-formulaires

Les différents types de bannières ne peuvent pas être affichés dans un sous-formulaire. Cependant, vous pouvez définir et appeler le drawBanner dans un sous-formulaire et il sera affiché dans le formulaire principal. N'oubliez pas de mettre la variable image au niveau du formulaire principal. (voir le chapitre Prise en main)

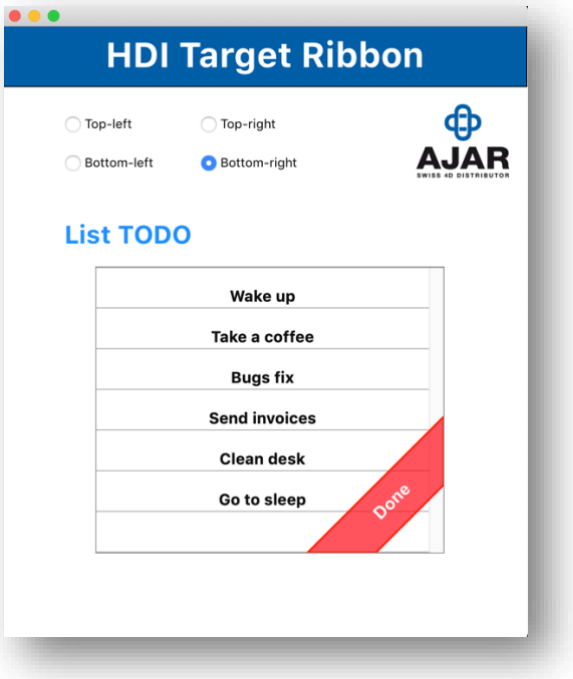
1.4 CornerRibbon – positionnement spécifique

Depuis la version 1.6 d'AJUI Banner, il est possible d'associer la bannière ruban directement à un objet de formulaire ou à une zone spécifique. Pour cela, deux nouvelles formules sont mises à disposition :

- **RibbonTargetName**
- **RibbonTargetCoordinates**

La première formule attend le nom d'un objet de formulaire. Le composant se chargera de récupérer les coordonnées de celui-ci, afin de positionner le ruban à l'un de ses quatre angles (top-left, top-right, bottom-left et bottom-right). Attention, l'objet de formulaire doit provenir du formulaire principal.

La seconde formule permet de passer directement les coordonnées (gauche, haut, droite, bas) afin de créer une zone qui recevra le ruban à l'un de ses quatre coins. Cette méthode permet de passer outre la limite de la première méthode en spécifiant par exemple des coordonnées qui correspondent à la position d'un objet de sous-formulaire (voir la méthode de 4D **CONVERT COORDINATES** pour convertir les coordonnées entre sous-formulaire et formulaire).



2 Méthodes du composant

2.1 New AJUI_Banner

Cette méthode va vous permettre de générer une instance de AJUI Banner sous forme d'objet. Cet objet va contenir l'ensemble des propriétés pouvant composer une bannière et les formules permettant de modifier les valeurs de celle-ci. Nous reviendrons sur chaque propriété et la formule liée à chacune dans le prochain chapitre.

2.1.1 Fonction : DrawBanner

C'est une formule de l'instance permettant de lancer la génération et le positionnement de la bannière.

On peut l'appeler en écrivant *MonInstance.DrawBanner( )*. Son but est de récupérer l'ensemble des propriétés contenu dans l'objet (l'instance de bannière) qui l'a appelé, ici *MonInstance*. Elle va ensuite générer une image à partir d'un SVG créé en se basant sur les différentes propriétés. Et pour finir, elle va déplacer la variable de type image au bon endroit dans le formulaire et y associer l'image générée.

2.2 Dialog_HDI_AJUI_Banner

Cette méthode HDI (How Do I) permet de faire appel à un formulaire partagé du composant. C'est un formulaire d'exemple que nous fournissons afin que vous ayez un aperçu du rendu du composant (les captures d'écran du chapitre 1 sont faites à partir du HDI).

2.3 AJUIB_Info

Cette méthode retourne l'information concernant la version de votre composant.

3 Listes des propriétés

Une bannière possède un ensemble de propriétés qui vont permettre de définir sa représentation dans le formulaire hôte.

Dans ce chapitre, nous allons passer en revue les différentes propriétés existantes accessibles par une formule faisant office de Setter et également de Getter si on ne leur passe aucun paramètre.

Toutes les formules pourront être appelées au premier niveau de l'objet.

3.1 Banner

Propriétés liées à la bannière se trouvant au deuxième niveau de l'objet : **MyBanner.**banner

Nom	Type	Description	Par défaut	Formule
name	string	nom de la bannière si celle ci est de type *windows* ou nom du cornerRibbon s'il est du type cornerRibbon.	AJUI_Banner_2use or AJUI_cr_2use	BannerName
bannerBGColor	string	Couleur de fond de la bannière. Il est possible de définir un taux d'opacité (%). Voir valeur par défaut	black:50	BannerBGColor
bannerHeight	longint	Hauteur de la bannière.	150	BannerHeight
bannerWidth	longint	Largeur de la bannière.	400	BannerWidth
borderColor	string	Couleur de la bordure de la bannière.	darkgray	BorderColor
borderSize	longint	Largeur de la bordure de la bannière.	0	BorderSize
cornerRadius	longint	Arrondi des angles de la bannière.	15	CornerRadius
position	string	Position de la bannière pour le type « cornerRibbon». Position possible : top-left | top-right | bottom-left | bottom-right	bottom-right	BannerPosition
type	string	Type de bannière à utiliser. Type possible : cornerRibbon | window | spinner | windowSpinner	cornerRibbon	BannerType

Banner – Formules et leurs paramètres

Nom de formule	Paramètre(s)
BannerName	- Nom de la bannière (string)
BannerBGColor (couleur)	- Couleur de fond (string)
BannerHeight (hauteur)	- Hauteur de bannière (longint)
BannerWidth (largeur)	- Largeur de bannière (longint)
BorderColor (couleur)	- Couleur de bordure (string)
BorderSize (taille)	- Taille de bordure (longint)
CornerRadius (arrondi)	- Arrondi des angles (longint)
BannerPosition (position)	- Positon bannière « cornerRibbon» (string)
BannerType (type)	- Type de bannière (string)

3.2 Text

Propriétés liées au contenu textuel se trouvant au deuxième niveau de l'objet : **MyBanner.**text

Nom	Type	Description	Par défaut	Formule
message	string	Message qui sera affiché dans la bannière	AJUI Banner	Message
textColor	string	Couleur du texte du message	White	TextColor
textFontName	string	Nom de la police. Certaines polices peuvent ne pas être compatible SVG	Arial	TextFontName
textFontSize	longint	Taille de la police	32	TextFontSize
textFontStyle	string	Propriété SVG pouvant être appliqué au niveau texte. Propriété possible : normal | italic | oblique | inherit	normal	TextFontStyle
textFontWeight	string	Propriété SVG pouvant être appliqué au niveau texte. Propriété possible : normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	bold	TextFontWeight

Text – Formules et leurs paramètres

Nom de formule	Paramètre(s)
Message (libellé)	- Message de la bannière (string)
TextColor (couleur)	- Couleur du texte (string)
TextFontName (police)	- Nom de la police (string)
TextFontSize (taille)	- Taille de la police (longint)
TextFontStyle (style)	- Style du texte (string)
TextFontWeight (épaisseur)	- Épaisseur du texte (string)

3.3 Window

Propriétés liées à la fenêtre se trouvant au deuxième niveau de l'objet : **MyBanner.**window

Nom	Type	Description	Par défaut	Formule
windowBGColor	string	Couleur de fond pour l'utilisation de la bannière « window ». Il est possible de définir un taux d'opacité (%). Voir valeur par défaut	black:50	WindowBGColor
windowRef	longint	Référence au formulaire devant recevoir la bannière. La valeur 0 représente le formulaire courant.	0	WindowRef

Window – Formules et leurs paramètres

Nom de formule	Paramètre(s)
WindowBGColor (couleur)	- Couleur de fond (string)
WindowRef (numéro)	- Référence de la fenêtre du formulaire (longint)

3.4 Picture

Propriétés liées à l'image au deuxième niveau de l'objet : **MyBanner.**picture

Nom	Type	Description	Par défaut	Formule
isPicture	booléen	Si vrai, la bannière contient une image, si non la bannière contient un texte. Ne concerne que la bannière « window »	False	IsPicture
picturePath	string	Chemin de fichier de l'image à afficher. L'image doit se trouver dans le dossier ressources de l'application 4D et le chemin est toujours relatif. Exemple : "Images/info.png" (*utilisés la constante folder separator de 4D pour construire la chaîne de façon propre*)	Chaîne vide	PicturePath

Picture – Formules et leurs paramètres

Nom de formule	Paramètre(s)
IsPicture (utiliser image)	- Utiliser image ou texte (booléen)
PicturePath (chemin de fichier)	- Chemin de fichier dans les ressources (string)

3.5 Spinner

Propriétés liées au « spinner » au deuxième niveau de l'objet : **MyBanner.banner.**spinner

Nom	Type	Description	Par défaut	Formule
scale	réel	Mise à l'échelle du « spinner » par rapport à la fenêtre ne s'applique pas pour le type « windowSpinner » car sa taille s'adaptera en fonction de la zone de texte.	0.5	SpinnerScale
side	string	Alignement du « spinner » par rapport au texte. Valeur possible : left | right	right	WindowSpinnerSide

Picture – Formules et leurs paramètres

Nom de formule	Paramètre(s)
SpinnerScale (échelle)	- Échelle du « spinner » (réel)
WindowSpinnerSide (alignement)	- Alignement du « spinner » (string)

3.6 Ribbon

Propriétés liées au « ribbon » au deuxième niveau de l'objet : **MyBanner.banner.**ribbon

Nom	Type	Description	Par défaut	Formule
targetName	longint	Nom de l'objet de formulaire qui recevra le ruban.	Chaîne vide	RibbonTargetName
TargetCoordinates	object	Coordonnées d'une zone qui recevra le ruban.	0 pour chaque coordonnées	RibbonTargetCoordinates

Picture – Formules et leurs paramètres

Nom de formule	Paramètre(s)
RibbonTargetName (nom)	- Nom de l'objet de formulaire (string)
RibbonTargetCoordinates (gauche, haut, droite, bas)	- Coordonnée gauche (longint) - Coordonnée haut (longint) - Coordonnée droite (longint) - Coordonnée bas (longint) En mode Getter, la formule retourne un objet contenant les quatre coordonnées.

4 Formules de gestion de la bannière

L'instance de bannière contient trois formules permettant de gérer le cycle de vie d'une bannière.

- **DrawBanner** : cette formule permet de lancer la génération de la bannière en se basant sur les propriétés et rendre l'objet de formulaire image recevant celle-ci visible.
- **HideBanner** : Cette méthode permet de cacher l'objet de formulaire image lié à l'instance. Elle exécutera également la formule « StopSpinner » en fonction du type de bannière de l'instance.
- **StopSpinner** : cette formule permet d'interrompre l'animation du « spinner » à l'aide d'un drapeau stocké dans le Storage de 4D. Elle décharge également l'objet de formulaire image afin d'éviter des effets de bord.

Attention : Les « Spinners » utilisent un drapeau stocké dans le Storage 4D. Faites attentions à ne pas supprimer inopinément cette propriété : *Storage.AJUIB_spinner.activate*.

5 Prise en main

AJUI Banner est un composant relativement simple à prendre en main si vous avez bien assimilé les différentes propriétés à disposition dans l'objet.

Pour cela, nous allons vous présenter succinctement trois extraits de code vous permettant la création de bannières.

Dans tous les cas, il vous faut ajouter un objet formulaire qui contiendra la bannière et qui devra remplir les conditions suivantes :

- Cet un formulaire objet associé à une variable de type image.
- Il doit obligatoirement s'appeler *AJUI_Banner* sur la première page de votre formulaire.
- S'il est sur une autre page du formulaire, ajouter le numéro de page au nom de celui-ci (exemple *AJUI_Banner2* pour la page 2, *AJUI_Banner3* pour la page 3, etc.).
- Généralement l'objet formulaire est placé au premier plan sauf cas spécifique en fonction de vos besoins.

La gestion de la visibilité de l'image n'est pas pris en charge par le composant. Nous vous conseillons de définir dans les propriétés l'objet formulaire à invisible par défaut et de le rendre visible quand cela est nécessaire. Nous vous recommandons également d'éviter d'utiliser l'objet en page zéro (même s'il est possible de le faire) car les objets formulaires des autres pages s'afficheront en juxtaposition, au dessus.

Cas 1 : Bannière « Window » avec texte

```
Form.banner:=New AJUI_Banner
Form.banner.BannerType("window")
Form.banner.IsPicture(False)
Form.banner.Message("You're welcome")
Form.banner.DrawBanner()
```

Cas 2 : Bannière « Window » avec image

```
Form.banner:=New AJUI_Banner
Form.banner.BannerType("window")
Form.banner.IsPicture(True)
Form.banner.PicturePath("Images"+Folder_separator+"info_white.png")
Form.banner.DrawBanner()
```

Cas 3 : Bannière « cornerRibbon » positionnée en bas à droite

```
Form.myBanner:=New AJUI_Banner
Form.myBanner.BannerType("cornerRibbon")
Form.myBanner.Message("Corner Ribbon")
Form.myBanner.BannerPosition("bottom-right")
Form.myBanner.DrawBanner()
```

Libre à vous ensuite de tester les différentes possibilités de construction des bannières et de les modéliser à votre convenance.

Rappel sur les restrictions *: Le composant prévoit que le texte tienne sur une seule ligne, donc prenez soin de bien adapter la taille de la bannière à la taille de votre texte.*

Concernant les types « spinner » et « windowSpinner », il vous suffit de reproduire ce que vous avez fait pour le type « window » en modifiant juste le nom du type. Essayer également de changer l'échelle pour le « spinner » et de modifier l'alignement pour « windowSpinner ». N'oublier pas d'utiliser la formule « StopSinner » pour stopper les animations.

6 Conclusion

Le but de ce document était de vous présenter les principes théoriques du composant ainsi que les différentes méthodes, formules et propriétés à votre disposition pour pouvoir générer des bannières.

Quant aux éléments pratiques présentés ils ont pour but de vous permettre de mettre le pied à l'étrier et d'aborder quelques cas particuliers qui pourrait intervenir dans l'utilisation du composant.

Vous désirez une aide pour l'implémentation du composant AJUI Banner dans votre application. Vous désirez modifier ou étendre ses fonctionnalités pour un usage spécifique. Vous désirez disposer du code source du composant AJUI Banner afin de pérenniser son usage dans votre application avec les futures versions de 4D. N'hésitez pas à nous contacter pour en discuter.