



# AJUI Button

# Manuel d'utilisation

<b>1</b>	<b>Introduction.....</b>	<b>4</b>
1.1	Qu'est-ce que AJUI Button.....	4
1.2	Composition d'un bouton.....	4
1.3	Types de bouton .....	5
1.3.1	Anatomie des boutons .....	5
1.3.2	Dimensionnement d'un bouton .....	8
1.4	Les états d'un bouton.....	8
1.4.1	Les événements et callbacks .....	9
1.4.2	La notion d'exception et les constantes du composant .....	9
1.5	Cycle de vie d'un bouton .....	11
1.5.1	Phase 1 : initialisation et paramétrage .....	11
1.5.2	Phase 2 : Calculs et génération du bouton .....	11
1.5.3	Phase 3 : Affichage et masquage .....	11
<b>2</b>	<b>Méthodes du composant.....</b>	<b>12</b>
2.1	New AJUI_Button.....	12
2.2	AJUI_Btn_LoadTemplates.....	12
2.3	AJUI_Btn_info.....	12
<b>3</b>	<b>Listes des propriétés.....</b>	<b>13</b>
3.1	Global.....	13
3.2	Box.....	14
3.3	Border .....	15
3.4	Text .....	16
3.5	Picture .....	17
3.6	Composite .....	19
<b>4</b>	<b>Fonction membre non liée à une propriété .....</b>	<b>20</b>
<b>5</b>	<b>Prise en main .....</b>	<b>21</b>
5.1	Installation .....	21
5.2	Principes d'utilisation.....	21
5.2.1	Prérequis .....	21
5.2.2	Base de création d'un bouton .....	21
<b>6</b>	<b>FAB (Floating Action Button).....</b>	<b>24</b>
6.1	Liste des propriétés.....	25
6.2	Positions prédéfinies.....	26
6.3	Fonctions membres.....	27
6.4	Mise en place.....	27
<b>7</b>	<b>Conclusion .....</b>	<b>27</b>

Version Control	Date	Commentaire (Changement)	Auteur
<b>1.0</b>	19.08.2019	Première version	Gary Criblez
<b>1.2</b>	28.11.2019	Nouvel état « Focus »	Gary Criblez
<b>1.3</b>	19.12.2019	FAB (Floating Action Button)	Gary Criblez

# 1 Introduction

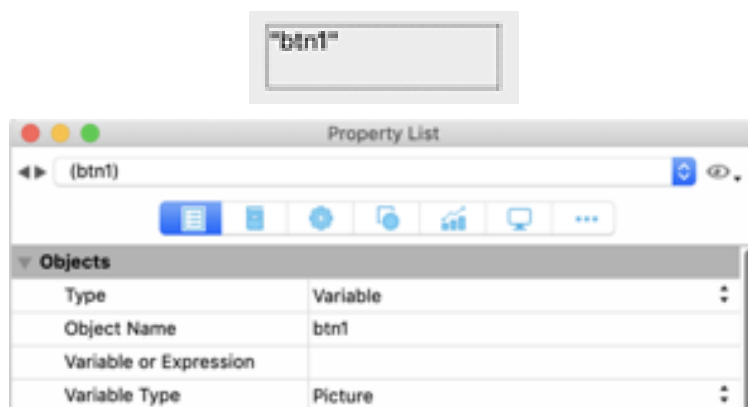
## 1.1 Qu'est-ce que AJUI Button

AJUI Banner est un composant développé dans le langage 4D. Son utilisation est destinée aux développeurs 4D. Il permet de générer et afficher dynamiquement des boutons d'interface dans le contexte d'un formulaire.

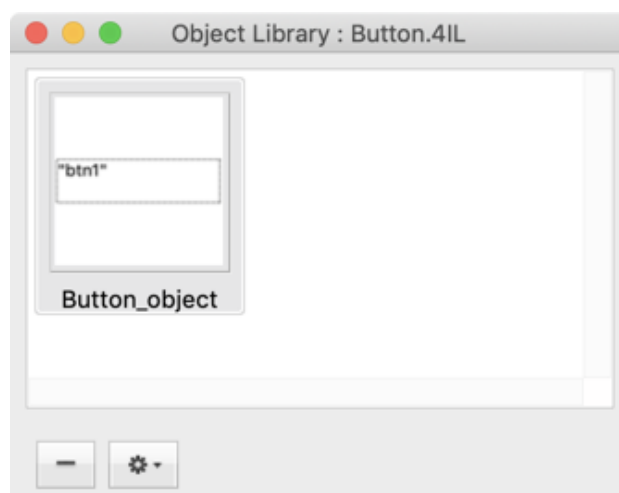
Un bouton est défini à l'aide d'une variable objet représentant une instance d'AJUI Button. Cette variable contient un ensemble de propriétés et de formules permettant de définir la structure et le contenu d'un bouton et de le générer dans le formulaire.

## 1.2 Composition d'un bouton

Fondamentalement, le bouton généré par AJUI\_Button est une variable objet de formulaire de type image que vous pouvez créer vous-même dans le contexte d'un formulaire.



Par rapport à ce point, nous vous mettons à dispositions un modèle prédéfini de cette variable objet de formulaire contenant les propriétés appropriées pour la création d'un AJUI Button. Elle est fournie dans un fichier de librairie d'objet « Button.4dlibrary » (la librairie est fournie en version projet pour la v18).



## 1.3 Types de bouton

AJUI Button offre deux principaux types de boutons

- Bouton simple
- Bouton composite

Un bouton simple est un rectangle dont les coins peuvent être arrondis ou un cercle comprenant un texte **OU** une image.

Un bouton composite est un rectangle dont les coins peuvent être arrondis ou un cercle comprenant un texte **ET** une image. L'image et le texte sont placés en fonction de la zone allouée à chacun, mais on peut également leur appliquer un padding entre eux. Dans ce cas, on parle de bouton composite lié. On assigne une position à l'image (left, right, top, bottom). Le texte lui se trouvera à l'opposé de cette position.

### 1.3.1 Anatomie des boutons

#### Bouton simple

Un bouton simple inclut un Texte OU une image. Voici ses représentations graphiques et ses propriétés géométriques.

Texte :

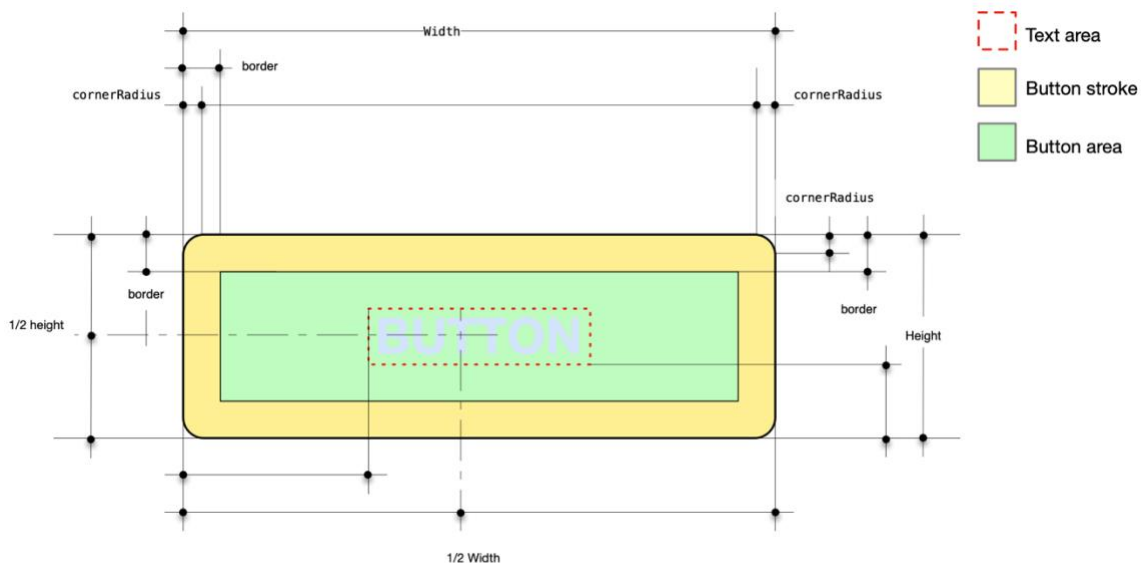
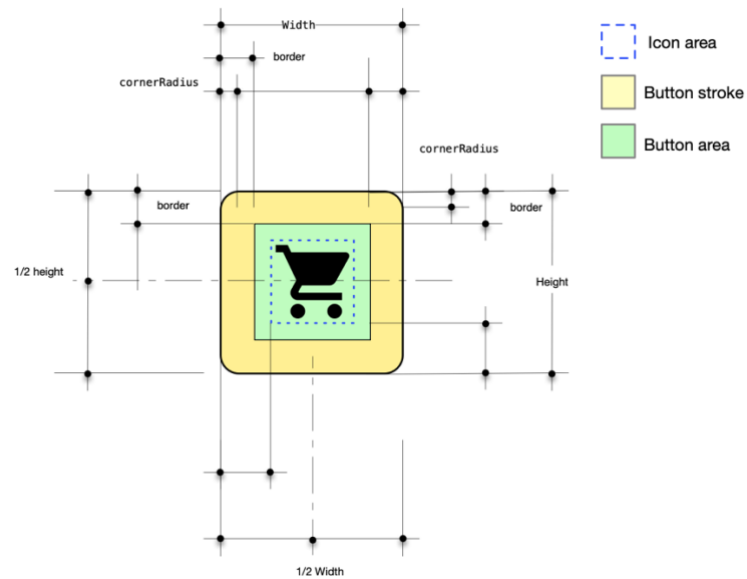


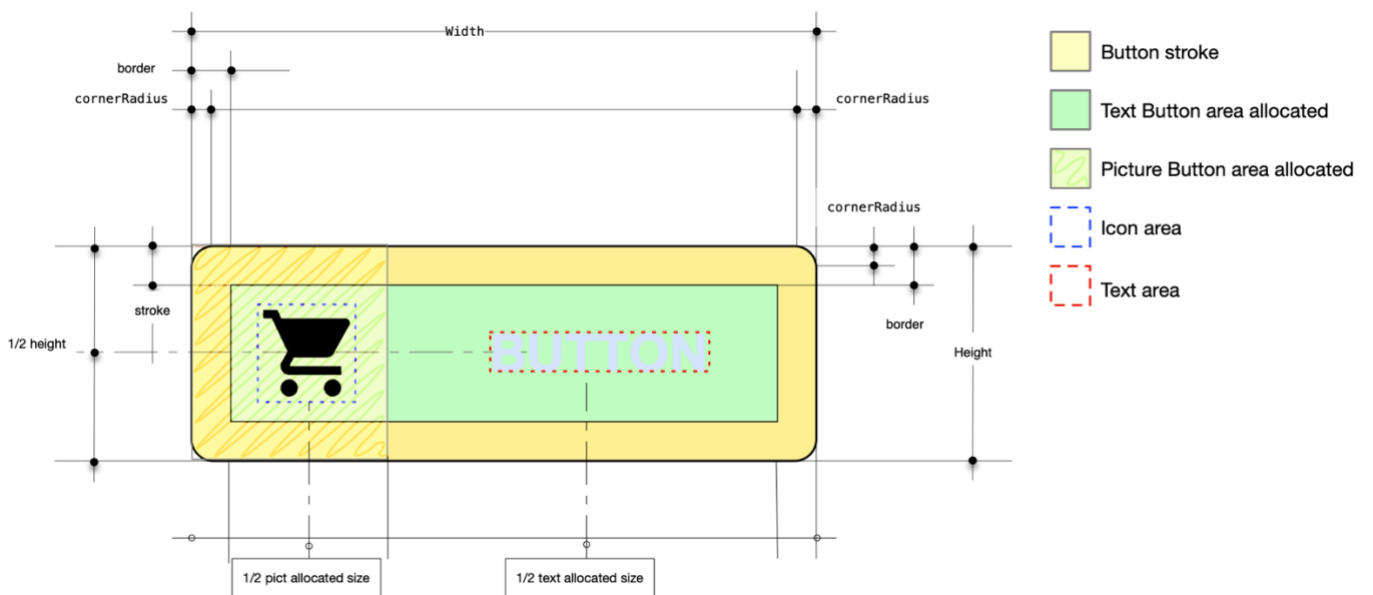
Image :



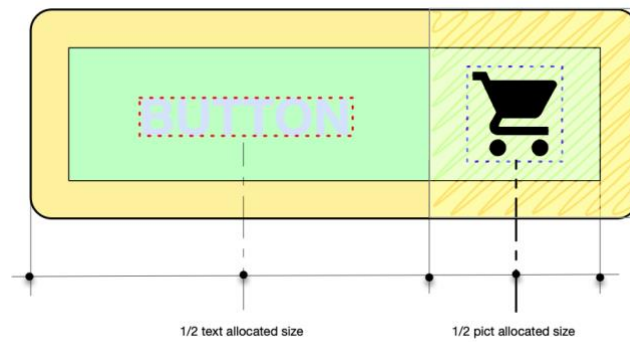
## Bouton composite

Un bouton composite inclut un Texte ET une image. Voici ses représentations graphiques et ses propriétés géométriques.

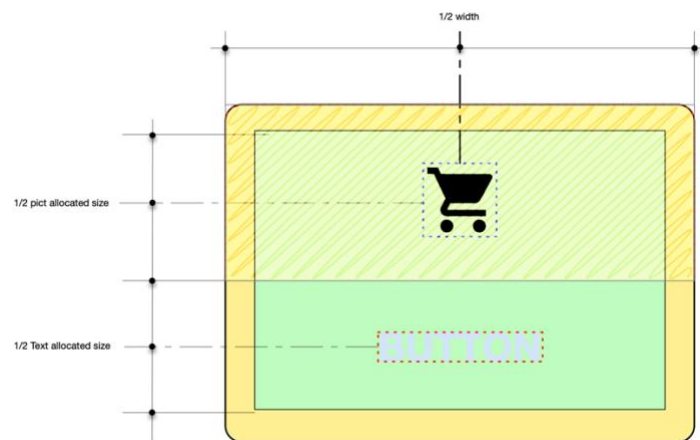
Position gauche :



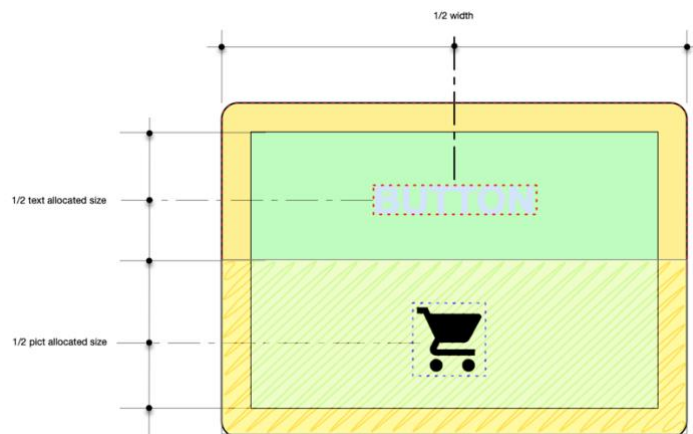
Position droite :



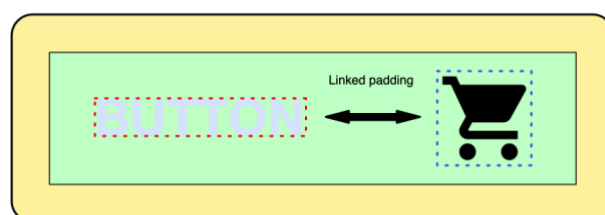
Position haut :



Position bas :



Lié :



### 1.3.2 Dimensionnement d'un bouton

Par défaut, le bouton généré par le composant aura les dimensions définies dans l'éditeur de formulaire. Cependant, il est possible de les redéfinir à l'aide des fonctions membres d'une instance d'AJUI Button.

La taille de l'image peut être définie via les fonctions membres, mais dans un souci de cohérence, le composant redimensionnera l'image si elle dépasse les limites du bouton ou de la taille allouée dans le cas d'un bouton composite.

Le texte sera automatiquement rétréci et terminé par « ... » s'il dépasse les limites du bouton ou de la taille allouée dans le cas d'un bouton composite.

## 1.4 Les états d'un bouton

Un AJUI Button peut prendre une apparence et un contenu différent en fonction de cinq états et des exceptions qui leur seront définis.

### L'état « Default »

C'est l'état basique d'un bouton lorsqu'un utilisateur n'interagit pas avec lui ou que celui-ci n'est pas désactivé.

### L'état « Hover »

Cet état intervient lorsque l'événement « On Mouse Enter » se déclenche et se termine sur l'événement « On Mouse Leave ». Concrètement, l'utilisateur est en train de survoler le bouton avec la souris.

### L'état « Active »

Cet état intervient sur l'événement « On Click » et se termine sur l'événement « On Mouse Up ». À noter qu'il a la priorité sur l'état « Hover ». Concrètement, l'utilisateur clique sur le bouton et l'état dure jusqu'au relâchement de celui-ci.

### L'état « Disable »

Cet état prend place lorsqu'on désactive le bouton à l'aide de la fonction membre « Enable » de l'objet. Lorsque le bouton est désactivé, seul cet état est pris en compte.

### L'état « Focus »

Cet état intervient sur l'événement « On Getting Focus » et se termine sur l'événement « On Losing Focus ».



### 1.4.1 Les événements et callbacks

Comme vous avez pu le constater par rapport aux états, le composant nécessite que plusieurs événements soit activés sur l'objet de formulaire image afin de pouvoir gérer les différents états. Voici la liste de ceux-ci :

- On Load
- On Click
- On Double Click (optionnel)
- On Mouse Enter
- On Mouse Leave
- On Mouse Up
- On Getting Focus
- On Losing Focus

Concernant les deux événements sur les clics, on peut leur associer des callbacks à l'aide des fonctions membres. De base, l'événement sur le double clic n'est pas nécessaire sauf si vous utiliser le callback.

**Attention** : Toutes méthodes utilisées comme callback doivent être partagées (propriétés de la méthode) avec le composant pour que celui-ci puisse l'appeler. Pour éviter qu'une erreur se produise pour ce genre de cas, le composant vérifiera si la méthode est bien partagée et si ce n'est pas le cas, il la partagera de lui-même lors de l'exécution d'un callback. De plus, le composant proposera de créer la méthode de callback si elle n'existe pas lors de l'utilisation des Setters pour les callbacks et également avant l'exécution d'un callback.

### 1.4.2 La notion d'exception et les constantes du composant


Afin de pouvoir modifier la structure et le contenu d'un bouton, nous avons mis en place un mécanisme d'exception dans le composant qui s'appliquera sur tous les états hormis « Default » qui lui conserve toutes les propriétés de base.

Pour cela, la majorité des fonctions membres (voir la liste des propriétés pour les détails) attendent en premier paramètre une constante. Chaque constante représente l'un des cinq états. Voici la liste :

- AJUI\_btn\_default (pas d'exception créée)
- AJUI\_btn\_hover
- AJUI\_btn\_active
- AJUI\_btn\_disable
- AJUI\_btn\_focus

En fonction de la constante que vous allez passer à une fonction membre, celle-ci va créer une exception dans votre instance d'AJUI Button qui prendra la place de la valeur courante (Default) lorsque l'état qui lui est lié est déclenché.

	hover (final values)	hover	default
bgColor	#CECECE	#CECECE	#D4E3FE
borderColor	Red	Red	Black
borderSize	2	2	0
fontColor	White	White	#0148AA
fontName	Arial		Arial
fontSize	12		12
fontStyle	bold	bold	None
Opacity	80	80	100
replacingColorSVG	none		none



Exemple d'affectation d'une couleur au texte du bouton pour chaque état :

```
$Mybtn.FontColor(AJUI_btn_default;"lightblue")
$Mybtn.FontColor(AJUI_btn_hover;"blue")
$Mybtn.FontColor(AJUI_btn_active"darkblue")
$Mybtn.FontColor(AJUI_btn_disable;"grey")
$Mybtn.FontColor(AJUI_btn_focus;"red")
```

## 1.5 Cycle de vie d'un bouton

### 1.5.1 Phase 1 : initialisation et paramétrage

La première phase consiste à définir les propriétés du bouton qui va être affiché. Pour cela, le composant offre une méthode « New AJUI\_Button » générant un objet de base représentant une définition par défaut. Libre ensuite à l'utilisateur d'utiliser les fonctions membres attachées à l'objet afin de personnaliser ces propriétés à sa guise.

```
Form.btn3:=New AJUI_Button
Form.btn3.Name("btn3")

//default
Form.btn3.BGColor(AJUI_btn_default;"lightgrey")
Form.btn3.Label(AJUI_btn_default;"BTN 3")
Form.btn3.BorderSize(AJUI_btn_default;2)

//hover
Form.btn3.BGColor(AJUI_btn_hover;"darkgrey")
Form.btn3.Label(AJUI_btn_hover;"On Hover")

//active
Form.btn3.BGColor(AJUI_btn_active;"grey")
Form.btn3.Label(AJUI_btn_active;"On Click")
```

Cette étape est très importante, car toutes les autres étapes se baseront sur ce qui a été défini dans l'objet.

### 1.5.2 Phase 2 : Calculs et génération du bouton

La deuxième phase qui est la plus conséquente pour le composant est la création du bouton basée sur sa définition. Durant cette phase, il va se charger de récupérer les propriétés de l'objet afin d'effectuer une série de calcul et de contrôle afin de générer/affecter les différents éléments qui vont composer cet ensemble qu'on nomme "le bouton". C'est également dans cette phase que les exceptions liées à l'état sont traitées. Si un callback est assigné et que l'événement correspond à celui-ci, il sera exécuté à la fin de la génération.

### 1.5.3 Phase 3 : Affichage et masquage

À la fin de la génération, l'image résultante est associée à l'objet de formulaire image. La gestion de l'affichage du bouton se fait à l'aide des fonctions membres « Show » et « Hide » ou à l'aide des méthodes 4D permettant de gérer les objets de formulaires.

## 2 Méthodes du composant

### 2.1 New AJUI\_Button

**New AJUI\_Button ( { template\_obj } ) -> button\_obj**

Cette méthode retourne une variable objet qui représente une instance d'AJUI Button. Elle contient l'ensemble des propriétés et leurs valeurs par défaut ainsi que les formules (fonctions membres) pour les manipuler. Il est possible de lui passer en paramètre un objet afin d'importer un template d'AJUI Button (fichier JSON). L'objet attendu comme propriétés :

- *templateName* : Corresponds au nom du fichier JSON à importer (template). Si le fichier n'est pas trouvé, la méthode retournera une nouvelle instance d'AJUI Button.
- *templatePath* (optionnel) : Vous pouvez préciser un chemin pour récupérer le fichier sinon, le composant cherchera dans le dossier par défaut se trouvant dans les ressources de la base hôte (.../Ressources/AJUI\_Button\_Templates/).

```
$template_obj:=New object("templateName";"MyTemplate.json";"templatePath";"C:/MesTemplates/")
$MyBtn:=New AJUI_Button ($template_obj)
```

### 2.2 AJUI\_Btn\_LoadTemplates

**AJUI\_Btn\_LoadTemplates ({cheminDeDossier}) -> templates\_name\_col**

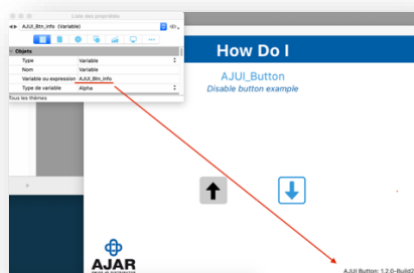
Méthode utilitaire permettant de récupérer l'ensemble des fichiers JSON d'un dossier dans une collection. Vous pouvez passer en paramètre le chemin du dossier contenant les templates. Si ce n'est pas le cas, la méthode cherchera dans chemin par défaut (.../Ressources/AJUI\_Button\_Templates/).

```
$folder:=Folder(Form.templateLocPref.path_global;fk_platform_path)
If ($folder.exists)
    $templates_col:=AJUI_Btn_LoadTemplates ($folder.platformPath)
End if
```

### 2.3 AJUI\_Btn\_info

**AJUI\_Btn\_info ( ) -> version\_txt**

Cette méthode retourne une chaîne de caractère représentant le numéro de version du composant.



### 3 Listes des propriétés

Un bouton possède un ensemble de propriétés qui vont permettre de définir sa représentation et ses interactions avec 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.

**Attention :** concernant les formules nécessitant l'état du bouton (state/état), celui-ci est obligatoire, seul le deuxième paramètre est optionnel pour réaliser un **Getter**.

Concernant les propriétés touchant aux couleurs, vous pouvez leur passer un code couleur en hexadécimal (Ex : 0x00FFFFFF) ou sous forme de chaînes CSS (Ex : "black", "#0A509E").

Il est recommandé de ne pas modifier manuellement les propriétés dont les valeurs ne sont pas définies par des formules, car elles sont gérées en interne par le composant.

NB : Dans les sections suivantes de ce chapitre 2 nous serons amenés à utiliser le terme **Btn (Button)** en lieu et place de **Bouton** pour des soucis de cohérences avec les noms définis en anglais dans les formules.

#### 3.1 Global

Propriétés globales se trouvant au deuxième niveau de l'objet : **btn.global**

Nom	Type	Description	Par défaut	Formule
composition	string	Composition du bouton : text, picture, composite.	text	Composition
name	string	Nom du bouton et donc de l'objet de formulaire image.	Chaîne vide	Name
onClickCB	string	Nom de la méthode de callback sur l'événement « On Click ». Vous pouvez passer en deuxième paramètre un objet contenant les paramètres du callback.	Chaîne vide	OnClick
onDoubleClickCB	string	Nom de la méthode de callback sur l'événement « On Double Click ». Vous pouvez passer en deuxième paramètre un objet contenant les paramètres du callback.	Chaîne vide	OnDoubleClick
type	string	Type du bouton (forme) : rectangle, circle.	rectangle	Type

#### Global – Formules et leurs paramètres

Nom de formule	Paramètre(s)
<b>Composition</b> ({composition})	- Composition du bouton (string)
<b>Name</b> ({nom})	- Nom du bouton (string)
<b>OnClick</b> ({callback {; paramètres}})	- Callback sur clic (string) - Paramètres (objet)
<b>OnDoubleClick</b> ({callback {; paramètre}})	- Callback sur double clic (string) - Paramètre (objet)
<b>Type</b> ({type})	- Type du bouton (string)

## 3.2 Box

Propriétés liées au conteneur se trouvant au deuxième niveau de l'objet : **btn.box**

Nom	Type	Description	Par défaut	Formule
bgPrimaryColor	string	Couleur du bouton principale. Il est possible de définir un taux d'opacité (%) (Ex: black:80 - noir avec une opacité de 80%)	#4d9ad4	BGColor
cornerRadius	longint	Valeur du rayon appliqué aux quatre angles du bouton.	5	CornerRadius
height	longint	Hauteur du bouton. La valeur -1 prend comme hauteur celle définie dans l'éditeur de formulaire	-1	Height
resize	booléen	Si vrai (True), il prendra en compte les valeurs définies (width et height). Si faux (False), il prend en compte les dimensions de l'objet de formulaire image défini dans l'éditeur de formulaire.	False	Resizable
width	longint	Largeur du bouton. La valeur -1 prend comme largeur celle définie dans l'éditeur de formulaire.	-1	Width

### Box – Formules et leurs paramètres

Nom de formule	Paramètre(s)
<b>BGColor</b> (état {; couleur})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Couleur de fond primaire (string)</li> </ul>
<b>CornerRadius</b> (état {; taille})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Taille de l'arrondi des bords (longint)</li> </ul>
<b>Height</b> (état {; largeur})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Largeur du bouton (longint)</li> </ul>
<b>Resizable</b> ({redimensionner})	<ul style="list-style-type: none"> <li>- Redimensionne l'objet formulaire (booléen)</li> </ul>
<b>Width</b> (état {; hauteur})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Hauteur du bouton (longint)</li> </ul>

### 3.3 Border

Propriétés liées à la bordure se trouvant au deuxième niveau de l'objet : **btn.border**

Nom	Type	Description	Par défaut	Formule
color	string	Couleur de la bordure. Il est possible de définir un taux d'opacité (%).	#0a509e	BorderColor
size	longint	Taille de la bordure.	0	BorderSize

#### Border – Formules et leurs paramètres

Nom de formule	Paramètre(s)
<b>BorderColor</b> (état {; couleur})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Couleur de bordure (string)</li> </ul>
<b>BorderSize</b> (état {; taille})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Taille de la bordure (longint)</li> </ul>

### 3.4 Text

Propriétés liées aux éléments textuels se trouvant au deuxième niveau de l'objet : **btn.text**

Nom	Type	Description	Par défaut	Formule
fontColor	string	Couleur du texte. Il est possible de définir un taux d'opacité (%).	white	FontColor
fontName	string	Nom de la police à utiliser. Plusieurs polices peuvent être entrées en les séparant par une virgule sans espace. Le composant recherchera la première police utilisable par rapport à votre OS.	Arial, Helvetica, MS Sans Serif	FontName
fontSize	longint	Taille du texte.	16	FontSize
fontStyle	string	Style du texte (Bold, Italic, Underline, Strikethrough).	Chaîne vide	FontStyle
label	string	Texte à afficher.	Label	Label
marginHorizontal	longint	Marge horizontale appliquée au texte permettant d'ajouter un écart lorsque celui-ci est aligné à gauche ou à droite. Si le texte est centré, cela permet de le compresser. La marge s'applique de chaque côté du texte.	0	TextMarginHorizontal

#### Text – Formules et leurs paramètres

Nom de formule	Paramètre(s)
<b>FontColor</b> (état {; couleur})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Couleur du texte (string)</li> </ul>
<b>FontName</b> (état {; police})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Nom de la police (string)</li> </ul>
<b>FontSize</b> (état {; taille})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Taille du texte (longint)</li> </ul>
<b>FontStyle</b> (état {; style})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Style du texte (string)</li> </ul>
<b>Label</b> (état {; libellé})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Libellé du bouton (string)</li> </ul>
<b>TextMarginHorizontal</b> (état {; marge})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Marge horizontale (longint)</li> </ul>



## 3.5 Picture

Propriétés liées aux éléments images se trouvant au deuxième niveau de l'objet : **btn.picture**

Nom	Type	Description	Par défaut	Formule
colorToReplace	string	Couleur actuelle d'un SVG. Ici c'est la valeur de la balise « fill »	Chaîne vide	ColorToReplace
height	longint	Hauteur de l'image. La valeur -1 prend la hauteur originelle de l'image.	-1	PictureHeight
opacity	longint	Taux d'opacité de l'image.	100	PictureOpacity
path	string	Chemin de dossier depuis le dossier « ressources ».  Vous pouvez également utiliser « # » plus le nom de votre fichier image comme chemin (valable uniquement avec un template AJUI Button importé si la ou les images associées sont dans le même dossier que le template)	Chaîne vide	PicturePath
ratio	réel	Agrandis/réduis la taille de l'image. Il est pris en considération à la place des propriétés hauteur et largeur. La valeur 1 désactive la prise en compte du ratio (1 = taille actuelle de l'image).	1	PictureRatio
replacingColor	string	Couleur devant remplacer la couleur actuelle dans une image de type SVG.	Chaîne vide	ReplacingColor
scale	longint	Redimensionnement de l'image par rapport à la taille du bouton d'après ce pourcentage. À la priorité sur toutes les autres propriétés modifiant la taille. La valeur zéro désactive cette opération. La valeur doit être comprise entre 0 et 100	0	PictureScale
width	longint	Largeur de l'image. La valeur -1 prend la largeur originelle de l'image.	-1	PictureWidth

## Picture – Formules et leurs paramètres

Nom de formule	Paramètre(s)
<b>ColorToReplace</b> (état {; couleur})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Couleur à remplacer (string)</li> </ul>
<b>PictureHeight</b> (état {; hauteur})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Hauteur de l'image (longint)</li> </ul>
<b>PictureOpacity</b> (état {; pourcentage})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Opacité de l'image (longint)</li> </ul>
<b>PicturePath</b> (état {; chemin})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Chemin du fichier image (string)</li> </ul>
<b>PictureRatio</b> (état {; ratio})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Ratio de la taille de l'image (réel)</li> </ul>
<b>ReplacingColor</b> (état {; couleur})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Nouvelle couleur (string)</li> </ul>
<b>PictureScale</b> (état {; pourcentage})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Échelle de l'image (longint)</li> </ul>
<b>PictureWidth</b> (état {; largeur})	<ul style="list-style-type: none"> <li>- État du bouton (constante)</li> <li>- Largeur de l'image (longint)</li> </ul>

## 3.6 Composite

Propriétés liées aux éléments composites se trouvant au deuxième niveau de l'objet : **btn.composite**

Nom	Type	Description	Par défaut	Formule
activeSecondColor	booléen	Active la couleur secondaire.	false	CompActiveSecondColor
bgSecondaryColor	string	Couleur secondaire pour le composite. Il est possible de définir un taux d'opacité (%).	#9fddf9	BGSecondaryColor
linked	booléen	Lie le texte à l'image et applique un padding entre eux (couleur secondaire n'est pas active dans ce cas).	false	CompLinked
padding	longint	Padding entre l'image et le texte si lié.	10	CompPadding
picturePosition	string	Position de l'image dans le bouton (ne s'applique que pour les composites). Position disponible : left, right, top, bottom	left	CompPicturePosition
pictSizeAllocation	longint	Espace alloué à l'image par rapport au texte. Cela sert à redimensionner l'image si elle dépasse les limites du bouton. Cette valeur sera appliquée sur la hauteur ou la largeur en fonction de la position de l'image par rapport au texte. Cette valeur représente des pixels.	30	CompPictSizeAllocation
textAlign	string	Alignement du texte dans le bouton composite (left, center, right).	center	CompTextAlign

### Composite – Formules et leurs paramètres

Nom de formule	Paramètre(s)
<b>CompActiveSecondColor</b> ({activer})	- Activer la couleur secondaire (booléen)
<b>BGSecondaryColor</b> (état {; couleur})	- État du bouton (constante) - Couleur de fond secondaire (string)
<b>CompLinked</b> ({lier})	- Lier le texte et l'image (booléen)
<b>CompPadding</b> (état {; padding})	- État du bouton (constante) - Padding entre le texte et l'image (longint)
<b>CompPicturePosition</b> (état {; position})	- État du bouton (constante) - Position de l'image (string)
<b>CompPictSizeAllocation</b> (état {; pourcentage})	- État du bouton (constante) - Taille allouée à l'image (longint)
<b>CompTextAlign</b> (état {; alignement})	- État du bouton (constante) - Alignement du texte (string)

## 4 Fonction membre non liée à une propriété

Liste des fonctions membres n'étant pas attachées à une propriété :

**Draw ( )** : génère le bouton correspond à l'état actuel.

**Enable (estActivé)** : rends l'objet de formulaire image actif/inactif et passe le bouton en état « disable » si inactif. Attends un booléen en paramètre. Lance automatiquement la méthode « Draw ».

**Export (nomTemplate { ; cheminDossierTemplate})** : Permet de créer un template de votre objet AJUI Button dans un fichier au format JSON. La formule requiert en premier paramètre le nom à associer au template (nom du fichier). Un deuxième paramètre optionnel vous permet de spécifier le chemin de dossier, sinon il sera exporté dans le dossier par défaut (.../Ressources/AJUI\_Button\_Templates/).

**Hide ( )** : cache le bouton.

**RemovePropertyException (formule ; état)** : permet de supprimer une exception liée à une propriété. Le nom de la formule liée à la propriété doit être mis en premier paramètre. Le deuxième paramètre correspond à l'état.

**RemovePropertyExceptions (formule)** : Permet de supprimer l'ensemble des exceptions liées à une propriété. Le nom de la formule liée à la propriété doit être mis en premier paramètre.

**ResetAllStates ( )** : Permet de supprimer l'ensemble des exceptions.

**ResetState (état)** : Permet de supprimer l'ensemble des exceptions d'un état. Vous devez passer la constante correspondant à l'état en paramètre (ne s'applique pas à l'état défaut).

**Show ( )** : affiche le bouton.

À noter que les fonctions membres **Enable**, **Hide** et **Show** sont en fait des wrapper des deux méthodes 4D « OBJECT SET VISIBLE » et « OBJECT SET ENABLE ». Vous pouvez très bien les utiliser directement si vous le souhaitez.

## 5 Prise en main

### 5.1 Installation

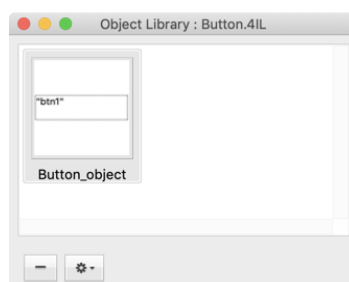
AJUI Button doit être placé dans le dossier composant de votre application.

### 5.2 Principes d'utilisation

Dans cette partie, nous décrivons la suite d'opération à réaliser afin de générer un simple Bouton dans le contexte du formulaire principal.

#### 5.2.1 Prérequis

AJUI Button a besoin d'un objet de type image qui va servir de modèle. Cet objet est fourni avec la librairie "Button.4IL" que vous pouvez ouvrir depuis votre application (/File/Open/Object Library...) et il vous suffira de faire glisser (Drag & Drop) l'objet « Button\_object » sur votre formulaire. Dans le cadre de la v18 de 4D, il vous faut utiliser la version projet de la librairie (seulement en mode projet).



Vous pourrez dupliquer cet objet pour créer autant de boutons que nécessaire. Chaque bouton devra avoir un nom spécifique que vous pouvez définir à votre fantaisie.

Si vous créez le bouton par vous-même, n'oubliez pas d'activer les événements requis au fonctionnement du bouton (voir le chapitre sur les événements).

#### 5.2.2 Base de création d'un bouton

La première chose à faire lors de la création d'un bouton est d'ouvrir la méthode de votre objet de formulaire image et de créer une instance avec la méthode « New AJUI\_Button ». Ensuite, associer le nom de votre bouton à l'instance avec la fonction membre « Name ». Voilà, le bouton est créé et va utiliser les valeurs par défaut, il ne vous reste plus qu'à adapter les paramètres de ce bouton à vos besoins en termes d'interface et de fonctionnement.

Pour cela, il suffit d'appeler les fonctions membres des propriétés qui sont décrites dans le chapitre correspondant. Tester, les différentes compositions et types. Changer également les valeurs des propriétés pour différent état en utilisant les constantes. Pour finir, encapsuler le tout dans un événement « on load » et lancer la fonction « Draw ».

Dans la suite du chapitre, quelques exemples sont exposés et nous vous conseillons grandement d'étudier le code source de l'application « AJUI\_ButtonLab » servant de démo au composant.

## Bouton simple avec texte

```

Case of
  : (Form event=On_Load)
    Form.btn1:=New AJUI_Button
    Form.btn1.Name("btn1")
    Form.btn1.Composition("text")

    //default definition
    Form.btn1.Label(AJUI_btn_default;"Simple Text Button")
    Form.btn1.BGColor(AJUI_btn_default;"white")
    Form.btn1.BorderColor(AJUI_btn_default;"#47A1F1")
    Form.btn1.BorderSize(AJUI_btn_default;2)
    Form.btn1.FontStyle(AJUI_btn_default;"none")
    Form.btn1.FontColor(AJUI_btn_default;"#47A1F1")

    //hover definition
    Form.btn1.Label(AJUI_btn_hover;"Btn Hover")
    Form.btn1.BGColor(AJUI_btn_hover;"#47A1F1")
    Form.btn1.BorderSize(AJUI_btn_hover;3)
    Form.btn1.FontStyle(AJUI_btn_hover;"Bold")
    Form.btn1.FontColor(AJUI_btn_hover;"white")

    //active definition
    Form.btn1.Label(AJUI_btn_active;"Btn Active")

End case

Form.btn1.Draw()

```

## Bouton simple avec image

```

Case of
  : (Form event=On_Load)
    Form.btn2:=New AJUI_Button
    Form.btn2.Name("btn2")
    Form.btn2.Type("circle")
    Form.btn2.Composition("picture")

    //default definition
    Form.btn2.PicturePath(AJUI_btn_default;"svg/si-glyph-dice-1.svg")
    Form.btn2.PictureHeight(AJUI_btn_default;20)
    Form.btn2.PictureWidth(AJUI_btn_default;20)

    //hover definition
    Form.btn2.PicturePath(AJUI_btn_hover;"svg/si-glyph-dice-2.svg")
    Form.btn2.BGColor(AJUI_btn_hover;"red")
    Form.btn2.PictureHeight(AJUI_btn_hover;30)
    Form.btn2.PictureWidth(AJUI_btn_hover;30)

    //active definition
    Form.btn2.PicturePath(AJUI_btn_active;"svg/si-glyph-dice-3.svg")
    Form.btn2.BGColor(AJUI_btn_active;"orange")
    Form.btn2.PictureHeight(AJUI_btn_active;40)
    Form.btn2.PictureWidth(AJUI_btn_active;40)

End case

Form.btn2.Draw()

```

**Bouton composite**

```

▼ Case of
  ▼ : (Form event=On_Load)
    Form.btn3:=New AJUI_Button
    Form.btn3.Name("btn3")
    Form.btn3.Composition("composite")
    Form.btn3.CompActiveSecondColor(True)

    //default definition
    Form.btn3.BGColor(AJUI_btn_default;"white")
    Form.btn3.Label(AJUI_btn_default;"default")
    Form.btn3.BorderSize(AJUI_btn_default;3)
    Form.btn3.FontColor(AJUI_btn_default;"#47A1F1")
    Form.btn3.PicturePath(AJUI_btn_default;"4D_v16_60px.png")
    Form.btn3.CompPicturePosition(AJUI_btn_default;"left")
    Form.btn3.CompPictSizeAllocation(AJUI_btn_default;60)

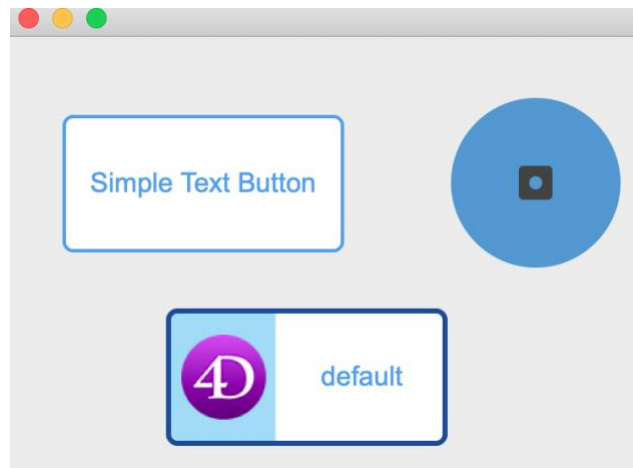
    //hover definition
    Form.btn3.BGColor(AJUI_btn_hover;"grey")
    Form.btn3.FontColor(AJUI_btn_hover;"white")
    Form.btn3.Label(AJUI_btn_hover;"hover")
    Form.btn3.FontStyle(AJUI_btn_hover;"Bold")

    //active definition
    Form.btn3.BGColor(AJUI_btn_active;"lightblue")
    Form.btn3.Label(AJUI_btn_active;"active")
    Form.btn3.BGSecondaryColor(AJUI_btn_active;"lightgrey")

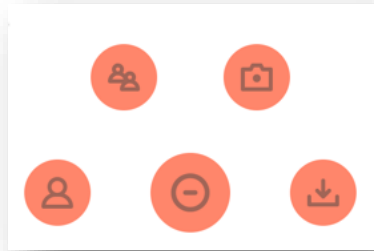
  End case

  Form.btn3.Draw()

```

**Résultat**

## 6 FAB (Floating Action Button)



Les FAB sont des boutons circulaires flottant au-dessus de l'interface utilisateur d'un formulaire. Ils agissent comme des boutons d'appel à l'action, censés représenter l'action unique que les utilisateurs exécutent le plus sur cet écran particulier.

La fonction FAB que nous vous proposons, vous permettra de créer et gérer ce type de boutons flottants.

Cette fonction permet de désigner une instance comme bouton principale qui va contenir une collection d'autres boutons secondaires.

Une fois les rôles définis, le bouton principal va être assigné à un callback interne au composant lui permettant d'afficher ou de masquer les autres boutons secondaires associés à la collection. Lors de l'affichage, les boutons secondaires vont être répartis autour du bouton principal reproduisant une forme de cercle. La taille, le sens et la répartition des boutons secondaires sur ce cercle peuvent être définis via les propriétés liées à la fonctionnalité FAB.

Il est possible d'appliquer une animation lorsque les boutons secondaires sont affichés ou masqués.

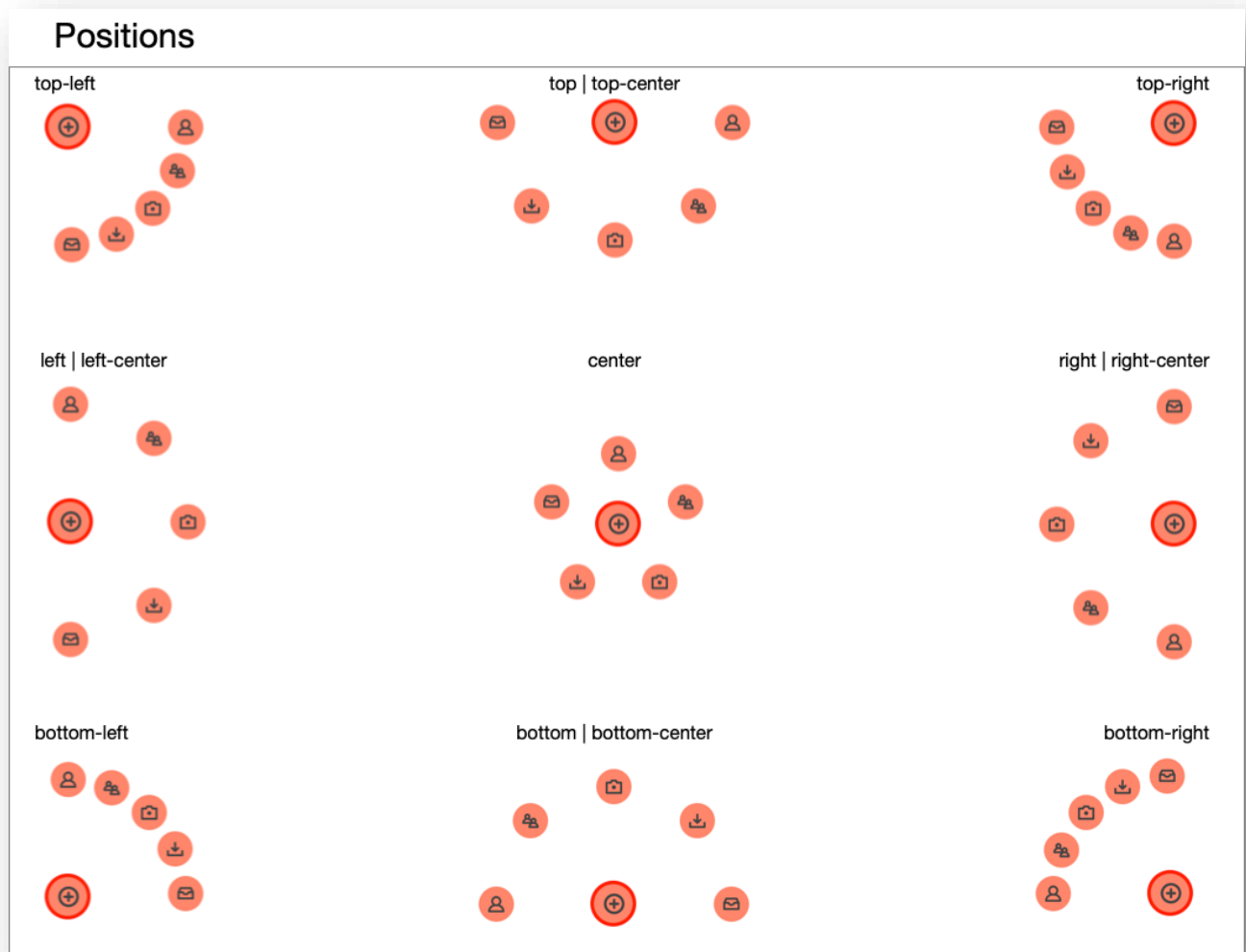


## 6.1 Liste des propriétés

Nom	Type	Description	Par défaut
mainBtn	booléen	Si vrai, le bouton est désigné comme bouton principal d'un système FAB. Il a comme unique fonction d'afficher et alternativement de masquer les boutons secondaires. Les éventuels callback définis (OnClick et OnDoubleClick) à ce bouton seront ignorés.	False
btnCol	collection	Collection des noms des boutons secondaires (nom de l'objet de formulaire image) qui vont composer le système FAB.  Leurs positions dans la collection peuvent être définies via une fonction d'ajout (FABAdd).	Collection vide
clockwise	booléen	Si vrai, la répartition des boutons se fait dans le sens des aiguilles d'une montre sinon c'est le contraire.	True
startAngle	longint	Angle à partir duquel le premier bouton secondaire sera positionné.  Les valeurs 0 à 359 sont admises. La valeur est en degré. Le point de départ des valeurs d'angle (valeur 0) est localisé à 3h.	270
angleToTravel	longint	Définit la taille relative de l'angle ou les boutons secondaires vont devoir être répartis en partant de l'angle de départ (startAngle) et en se basant sur le sens (clockwise).  Les valeurs 0 à 360 sont admises. La valeur relative est en degré.	180
radius	longint	Espace entre les boutons secondaires et le bouton principal. La valeur est exprimée en pixels relatifs au centre du bouton primaire et des boutons secondaires.	100
mainPosition	texte	Permet d'utiliser un positionnement de bouton prédéfini par le composant (voir le sous-chapitre « positions prédéfinies »).  L'utilisation d'une position prédéfinie va automatiquement mettre à jour les propriétés <i>startAngle</i> et <i>angleToTravel</i> .  Attention : la fonction membre liée à <i>mainPosition</i> agit comme un raccourci pour entrer des valeurs prédéfinies pour les deux paramètres ( <i>startAngle</i> et <i>angleToTravel</i> ).	Chaîne vide
animation	booléen	Permet d'activer l'animation du FAB	False

## 6.2 Positions prédéfinies

Voici les 9 positions prédéfinies et leurs identifiants que vous pouvez entrer dans la fonction membre "FABMainPosition".



## 6.3 Fonctions membres

Famille des fonctions membres liées à la fonction FAB.

- **FABIsMainButton** ( { \$active } ) : Getter/Setter pour la propriété « mainBtn ».
- **FABIsClockwise**( { \$clockwise } ) : Getter/Setter pour la propriété « clockwise ».
- **FABStartAngle** ( { \$angle } ) : Getter/Setter pour la propriété « startAngle ».
- **FABAngleToTravel** ( { \$angle } ) : Getter/Setter pour la propriété « angleToTravel ».
- **FABRadius** ( { \$radius } ) : Getter/Setter pour la propriété « radius ».
- **FABMainPosition** ( { \$namePosition } ) : Getter/Setter pour la propriété « mainPosition ».
- **FABAdd** ( \$nameBtn { ; \$position } ) : Ajoute le nom d'un bouton à la collection. Vous pouvez définir sa position dans liste en deuxième paramètre optionnel sinon il sera ajouté à la fin. Si le nom d'un bouton existe déjà dans la liste, la méthode agira comme une méthode de mise à jour de la position, et ne créera pas de doublon.
- **FABRemove** ( \$nameBtn ) : Retire de la collection le nom du bouton en paramètre.
- **FABClearCollection** ( ) : Supprime tous les éléments présents dans la collection.
- **FABIsAnimated** ( \$isActivated ) : Active/inactive l'animation

## 6.4 Mise en place

La première chose à faire pour mettre en place la fonction FAB consiste à créer chacun des boutons qui vont la composer.

Une fois réalisé, vous allez définir une instance comme bouton principal grâce à la fonction membre "FABIsMainButton". Ce bouton aura désormais comme unique fonction d'afficher et de masquer les boutons secondaires sur les événements « On Mouse Up » et/ou « On Double Click ». Les éventuels callback définis (OnClick et OnDoubleClick) pour ce bouton seront ignorés.

Ensuite, il faut ajouter à cette instance les noms des autres boutons avec la fonction "FABAdd", vous pouvez définir leur position dans la collection. Vous pouvez également activer l'animation du FAB.

Lors du chargement de votre formulaire, le bouton principal va automatiquement cacher les boutons secondaires. Pour un exemple concret d'implémentation, nous vous conseillons de regarder le code du formulaire HDI\_FAB dans le lab du composant.

## 7 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 gérer les boutons.

Quant aux éléments pratiques présentés, ils ont pour but de vous permettre de mettre un premier pied à l'étrier pour comprendre la mise en place du composant.

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