# Créer une application avec le générateur d’applications

## Objectif :

Créer une application

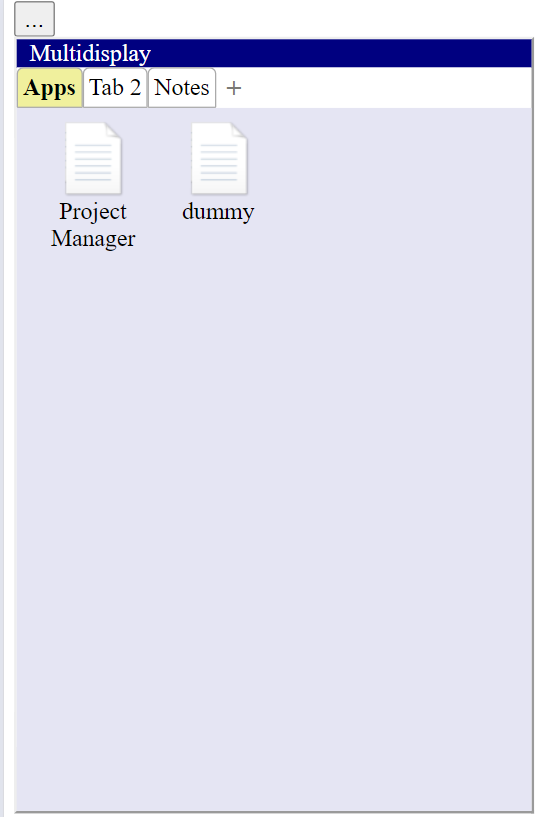
## Mise en place :

### Tutoriel

#### Création d’une application

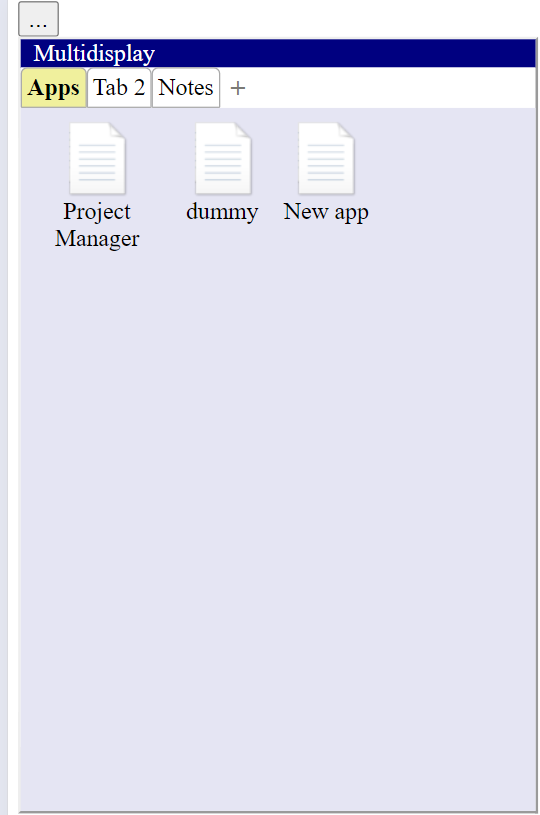
Dans le programme ***multidisplay***,

* Aller dans l’onglet ***Apps*** :

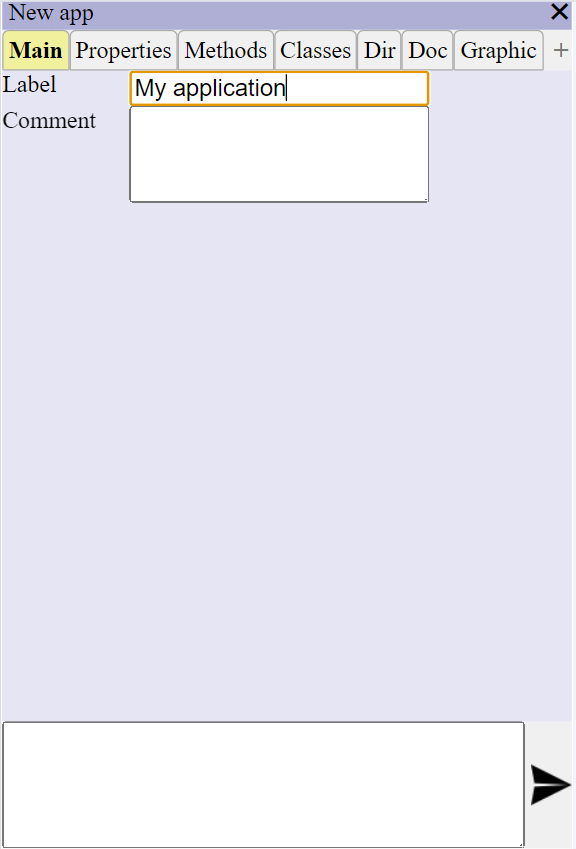


* Cliquer n’importe où dans la partie vide

Une nouvelle application apparaît :



* Cliquer dessus pour l’ouvrir :

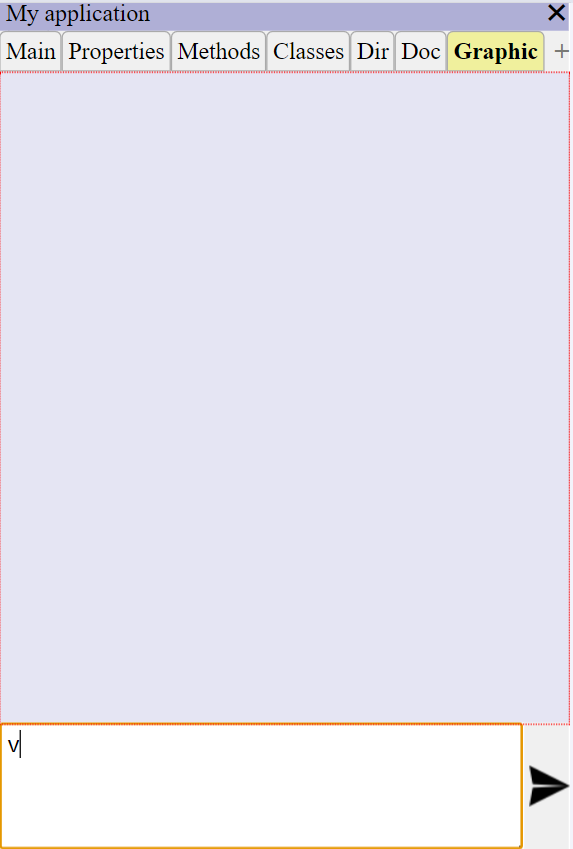


* Renseigner :
  + Son ***nom***

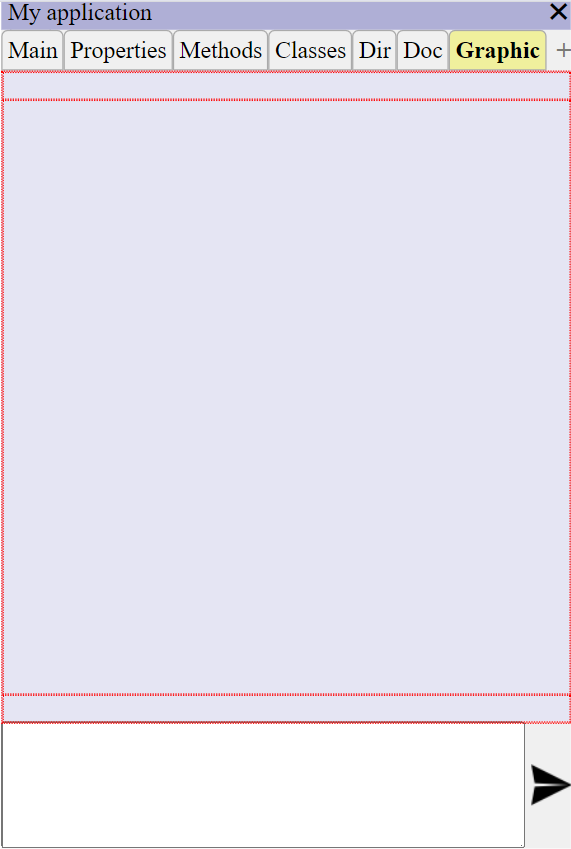
#### Création de l’interface graphique

Dans l’onglet ***Graphic***,

* Taper la commande ***v*** et appuyer sur la flèche :

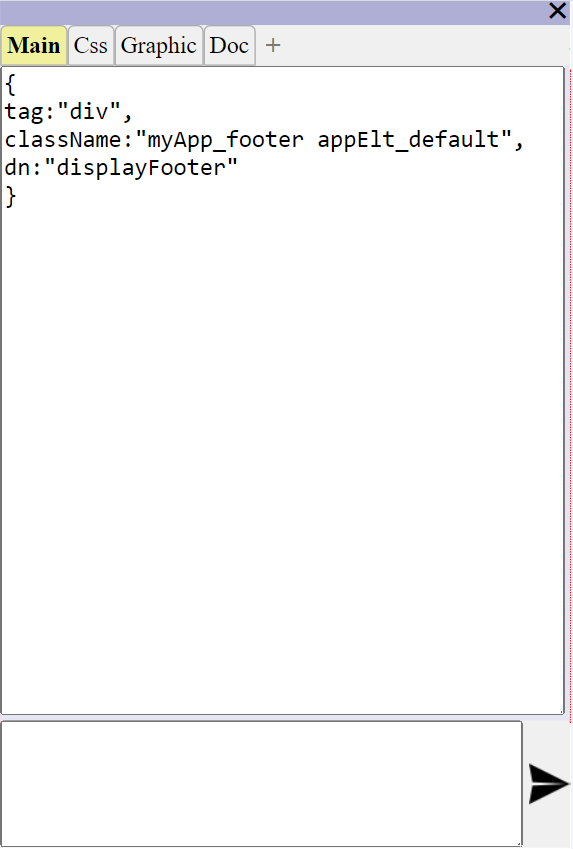


L’interface se scinde en 3 parties : le *header*, le *body* et le *footer* :



* Cliquer sur la partie du bas

Sa définition s’ouvre :



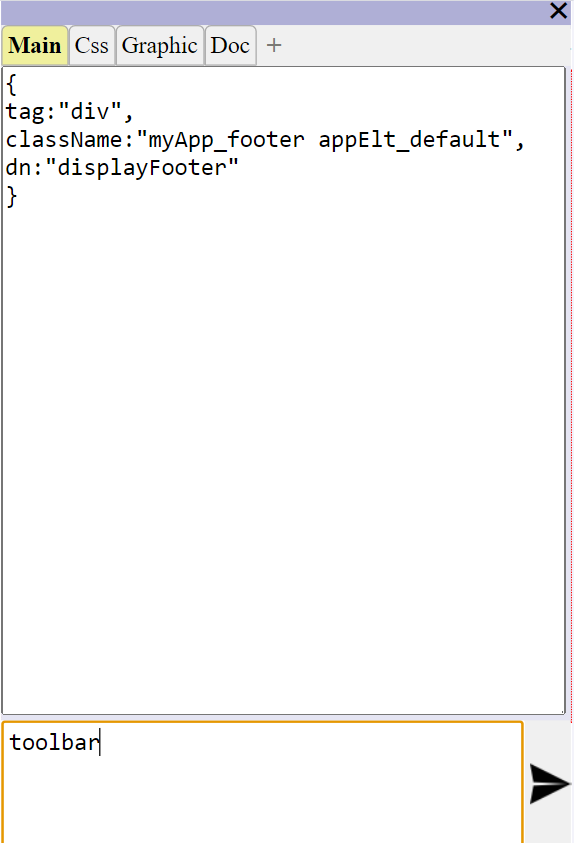
On comprend :

* Qu’elle correspond à un élément ***div*** qui a été créé automatiquement.
* Une ***classe*** lui a été affectée (créée à partir de la classe de l’élément parent concaténé à *\_footer*)
* Une deuxième classe lui a été affectée : *appElt\_default*. Elle permet de donner une taille d’affichage non nulle ainsi qu’un cadre en pointillé rouge afin qu’il soit visible par défaut. On peut la supprimer quand elle n’est plus nécessaire par la commande ***rs***.
* Une propriété ***dn*** lui a été ajoutée. Elle correspond au nom de la fonction qui lui sera affectée. Ainsi lors de la compilation une fonction portant ce nom sera générée. On peut changer ce nom par la commande *dn*.

#### Ajout d’une barre d’outils

Toujours dans le ***footer***,

* Taper la commande ***toolbar***

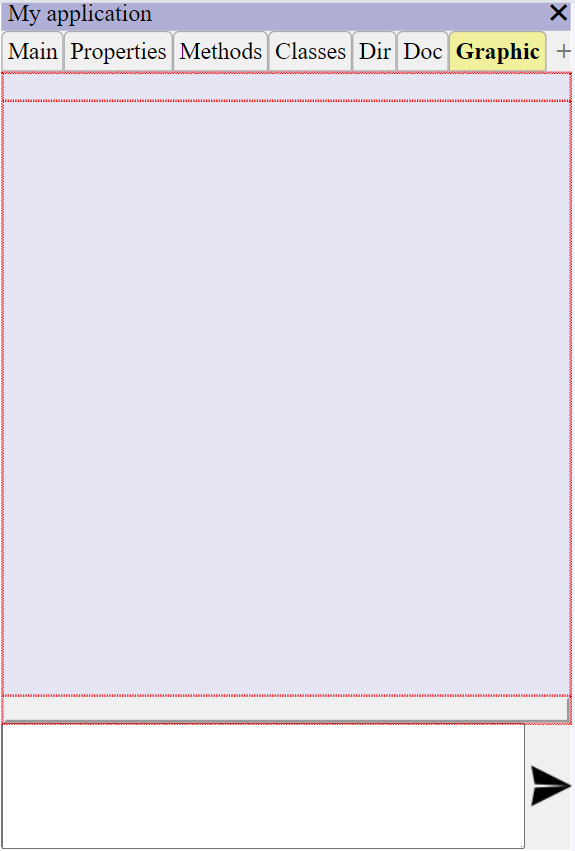


* Aller dans l’onglet ***Graphic*** :

Une barre d’outils apparaît à l’intérieur de l’élément div vu précédemment :

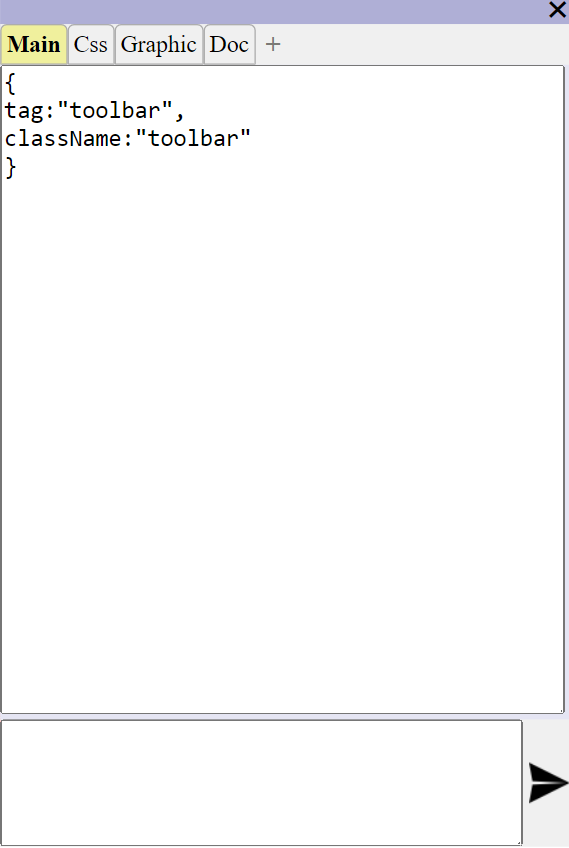


* Cliquer éventuellement sur la croix dans le coin supérieur droit pour fermer la définition du footer. Cela permet de mieux voir la barre d’outils :



* Cliquer sur la barre d’outils pour accéder à sa définition

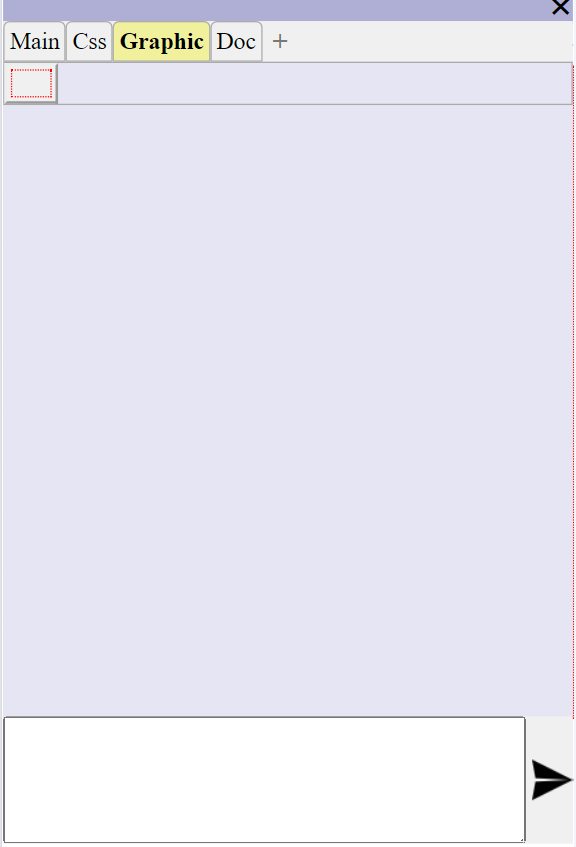
On voit qu’elle correspond à un élement *toolbar* :



Nous allons y ajouter des outils :

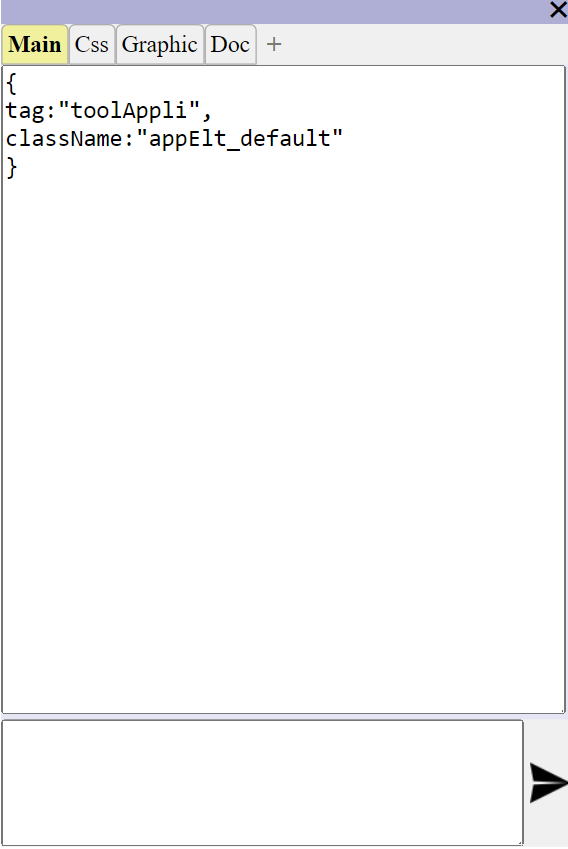
* Taper la commande ***apptool***
* Aller dans l’onglet ***Graphic*** :

L’outil a été ajouté :



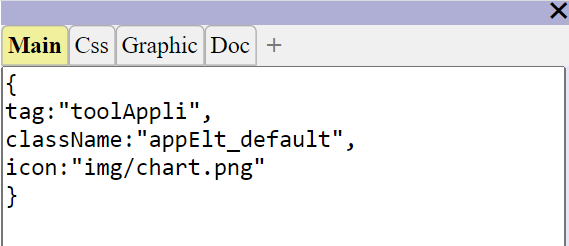
* Cliquer dessus pour accéder à sa définition :

Il s’agit d’un élément *toolAppli* :



* Taper la commande icon chart.png

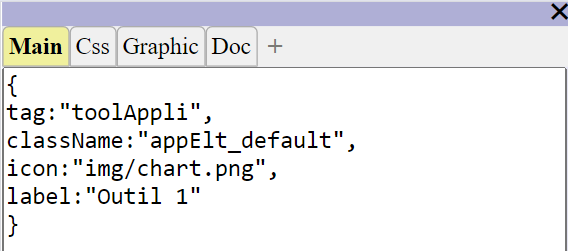
Une propriété ***icon*** a été ajoutée :



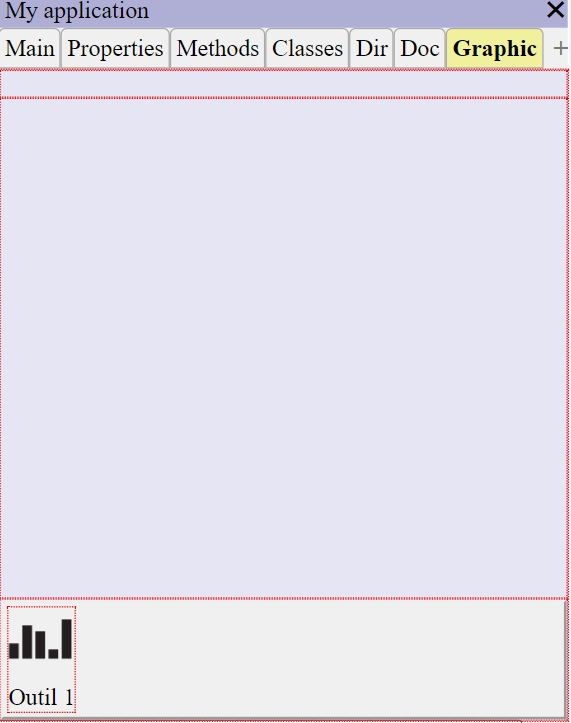
Remarquer qu’un **chemin** lui a été ajouté par défaut (*img/*) ainsi qu’une **extension**. Ces 2 élements sont en effet optionnels.

* Taper : label Outil 1

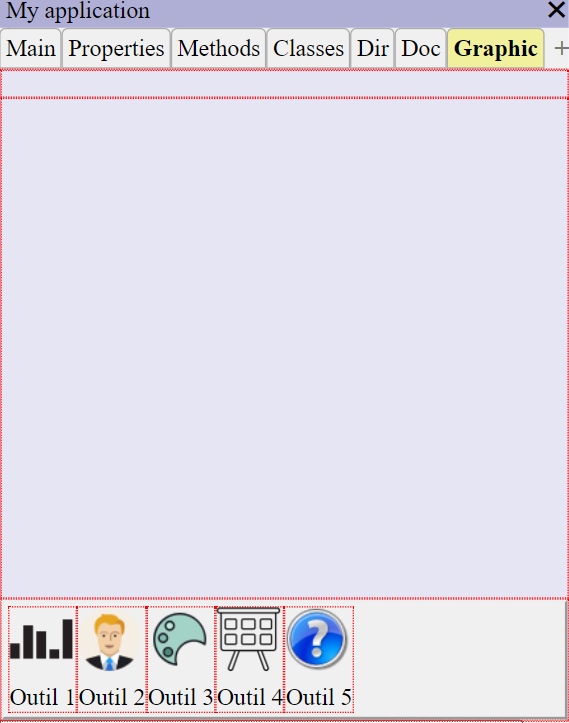
Il n’existe pas de commande *label*. Le système considère alors qu’il s’agit d’une **propriété** que l’on veut affecter



* Revenir à l’onglet ***Graphic*** au niveau général de l’application :



* Créer éventuellement d’autres outils de la même façon :



Ils apparaissent groupés sur la gauche. Pour répartir l’espace entre-eux,

* Aller dans la **toolbar**
* Taper la commande space



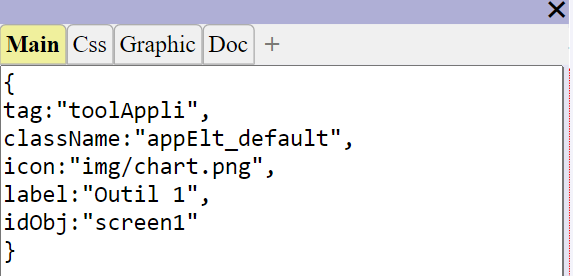
Nous souhaitons que le clic sur le premier outil ouvre un screen :



Pour rappel les éléments *toolAppli* prévoient une propriété nommée *idObj* qui permet d’appeler un tel objet. Il faudra que l’application aient créé cet objet à son lancement et l’ait rendu accessible par une de ses propriétés. On l’appellera ici *screen1*.

* Entrer dans l’*Outil 1*
* Taper idObj screen1

La propriété a été ajoutée :



Nous allons maintenant créer la classe qui permettra de créer cet objet screen1.

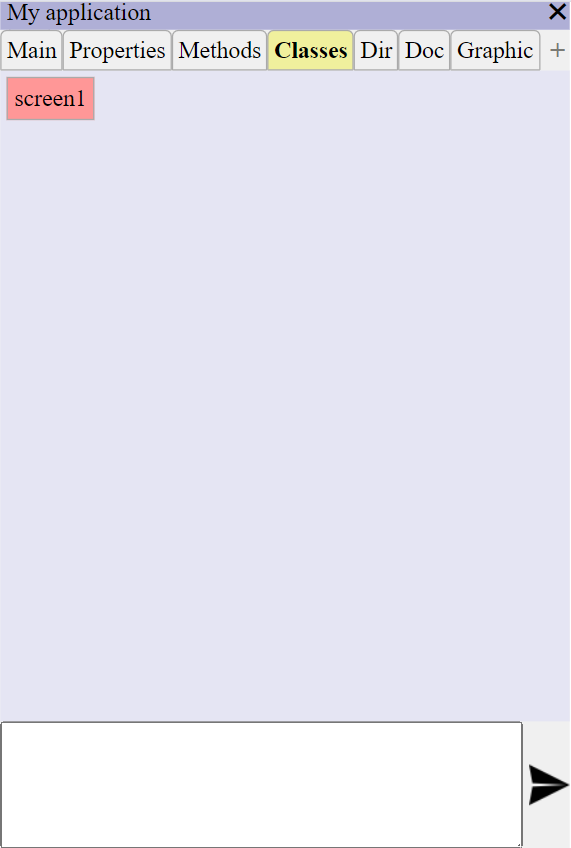
#### Création d’une classe

* Revenir au niveau de l’application
* Aller dans l’onglet Classes



* Taper la commande class screen1

La nouvelle classe est créé :



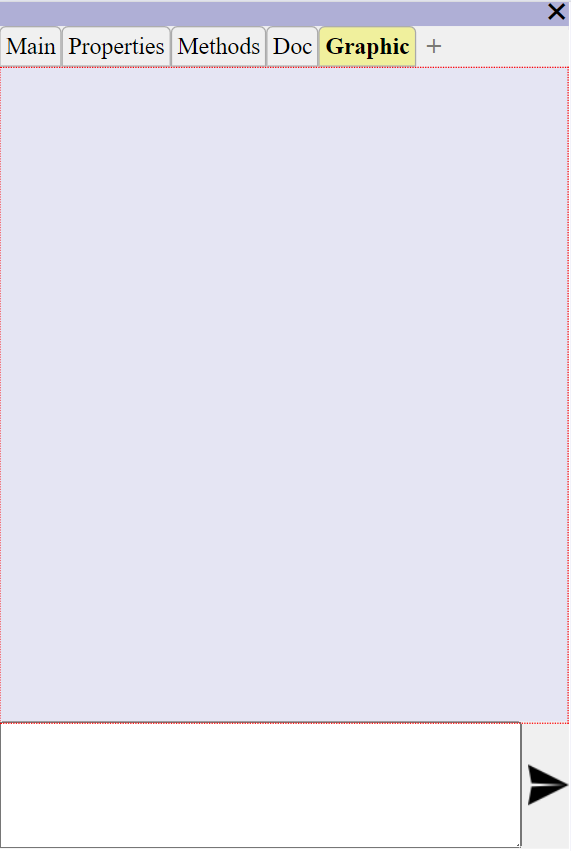
* Entrer dans sa définition
* Aller dans l’onglet ***Main*** :



* Renseigner :
  + ***Inherits*** : ***screen*** (dans la plupart des cas laisser la valeur par défaut : *graphic*)
* Aller dans l’onglet ***Graphic***

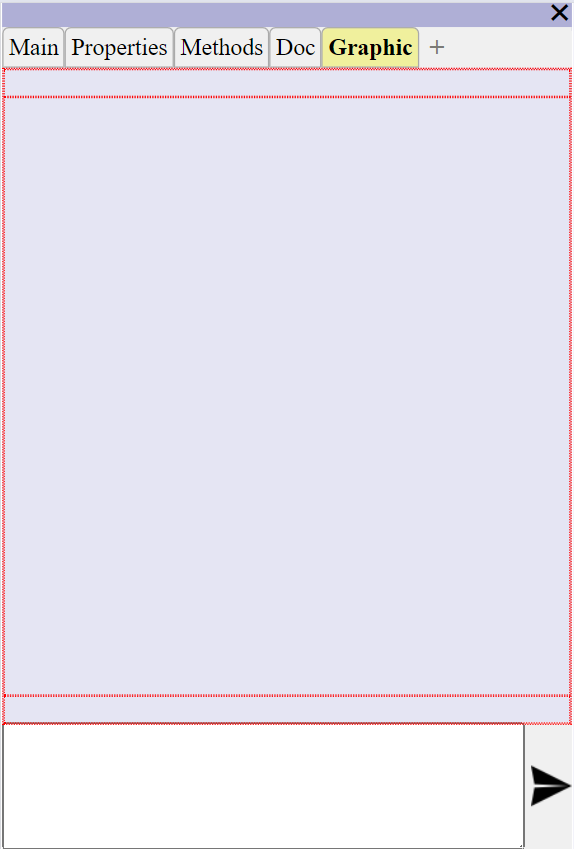
On voit que pour le moment il n’y a rien de prévu pour l’affichage (toutes les classes ne sont pas graphiques)

* Taper la commande **graphic** pour créer cet affichage :



Le contenu de cet onglet correspondra à la méthode display de la classe lors de sa compilation.

On peut alors y insérer les éléments désirés. Ici on créera les 3 éléments classiques (*header*, *body*, *footer*) par la commande ***v*** vue précédemment :

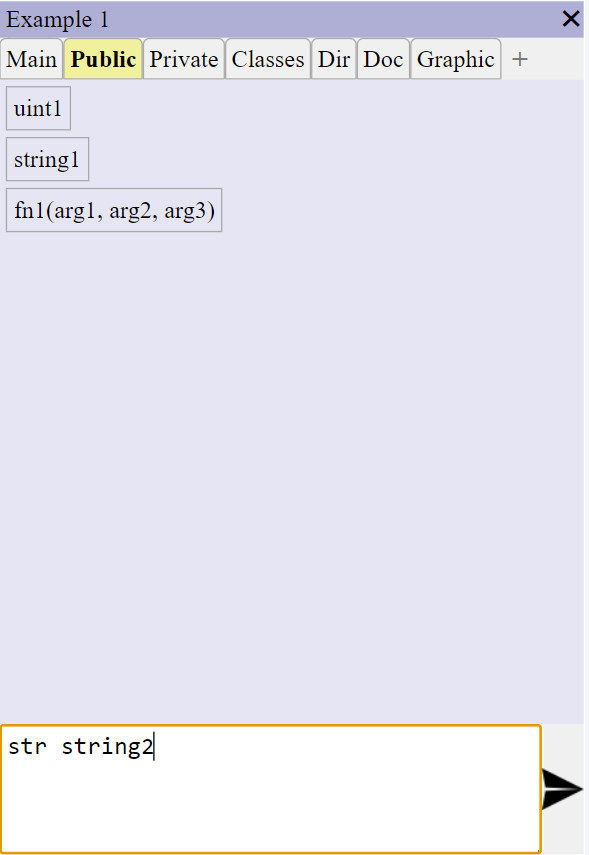


### Propriétés et méthodes

#### Création de propriété

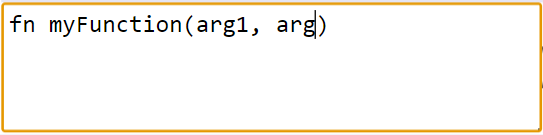
On utilise les commandes :

* ***int***
* ***uint***
* ***float***
* ***date***
* ***string*** (ou ***str***)
* ***boolean*** (ou ***bool***)
* ***array*** : Tableau []
* ***obj*** : Objet {}
* ***objarray*** (ou ***objarr***) : Tableau d’objets [{}]



#### Création de méthode

On utilise la commande ***fn***



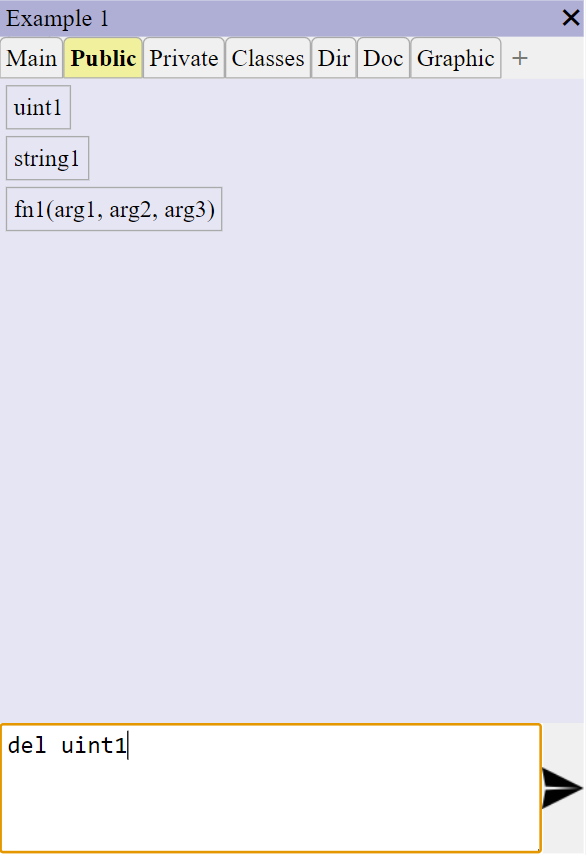
Elle apparaît en rouge, signifiant qu’elle n’est pas encore implémentée :



#### Suppression d’un membre

On procède par le menu contextuel : clic droit sur l’élement > ***Supprimer***

Ou on utilise la commande ***del***



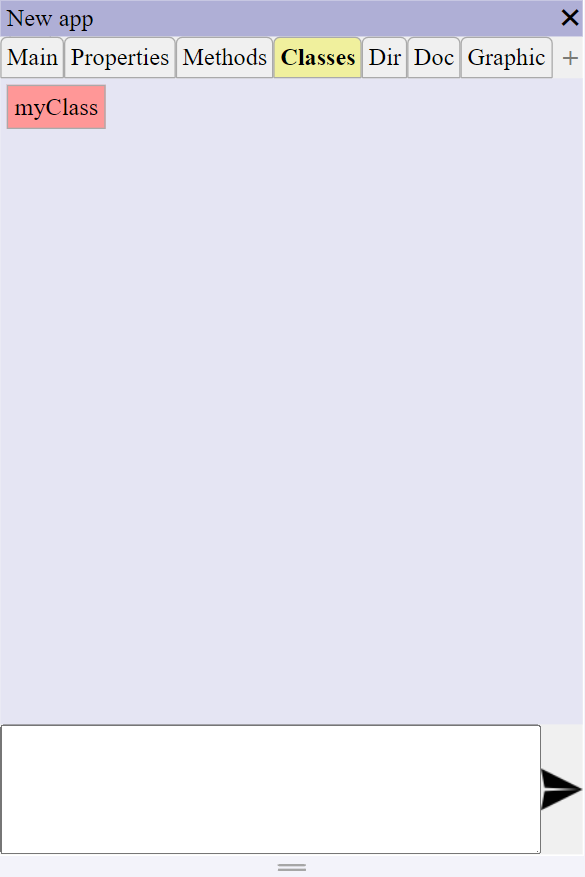
### Classes

#### Création

Commande ***class***



La nouvelle classe apparaît dans l’onglet ***Classes*** de l’application :

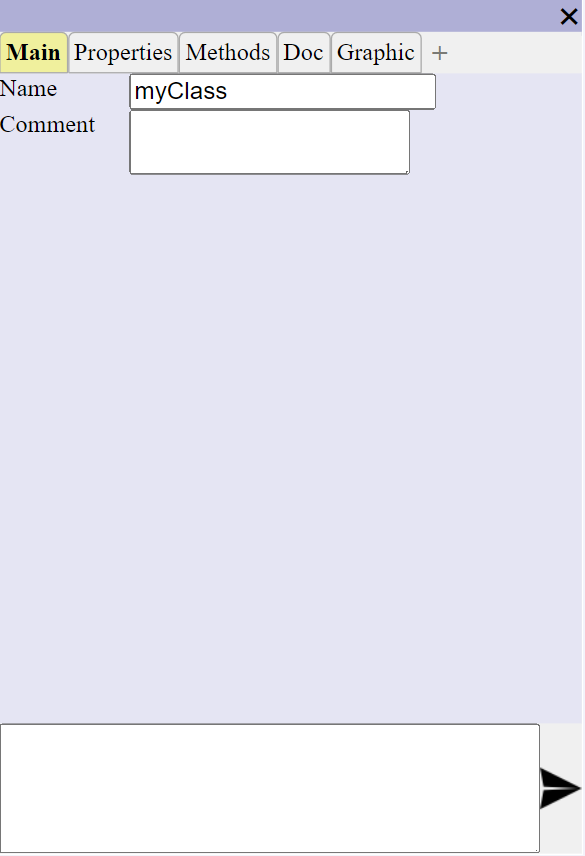


**Remarque**

Elle apparaît en fond rouge signifiant que pour le moment elle est vide.

* Cliquer dessus

Elle s’ouvre :



* On peut y créer des **propriétés** et des **méthodes** (voir *Propriétés et méthodes* de l’application)
* L’onglet *Graphic* est vide par défaut :



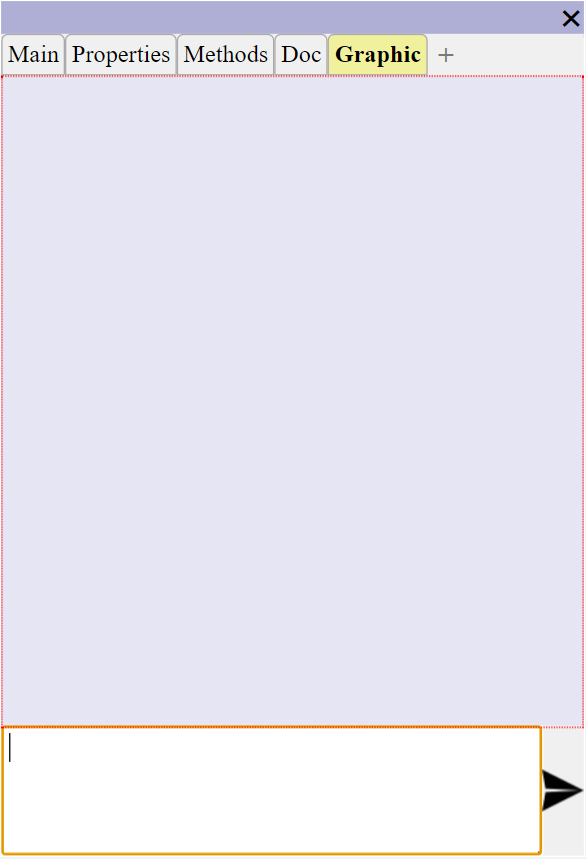
En effet une classe n’est pas nécessairement graphique.

#### Rendre une classe graphique

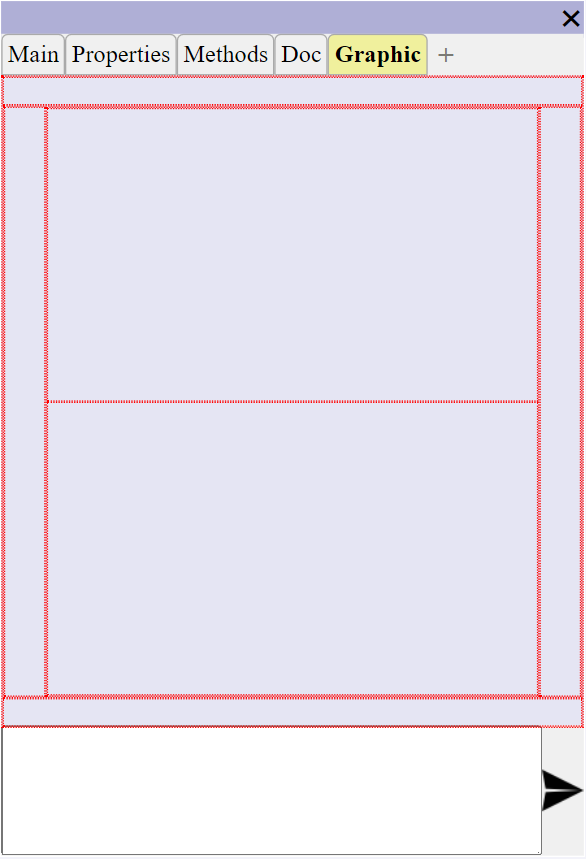
* Taper la commande ***graphic*** :



Un élement div apparaît encadré de rouge :

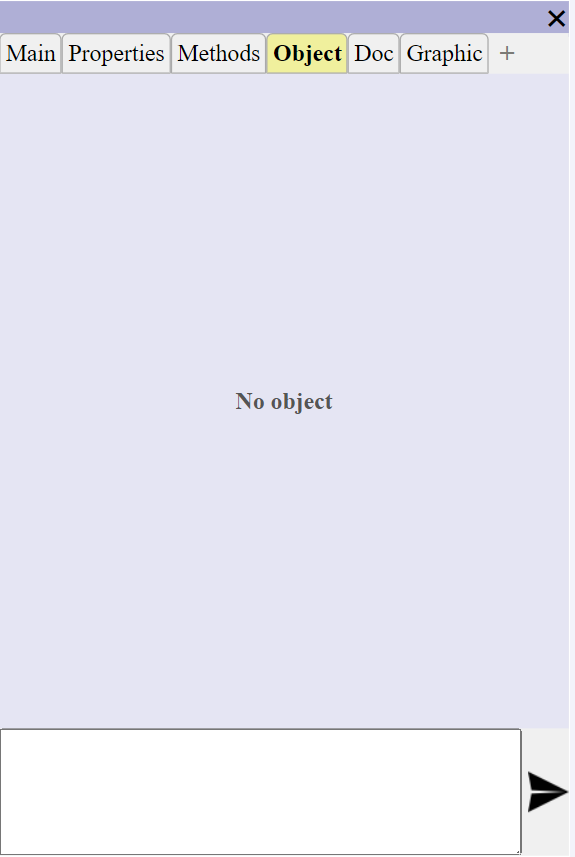


On peut alors modifier ses propriétés y insérer des insérer des éléments enfants :



### Lier un objet jsa à une classe

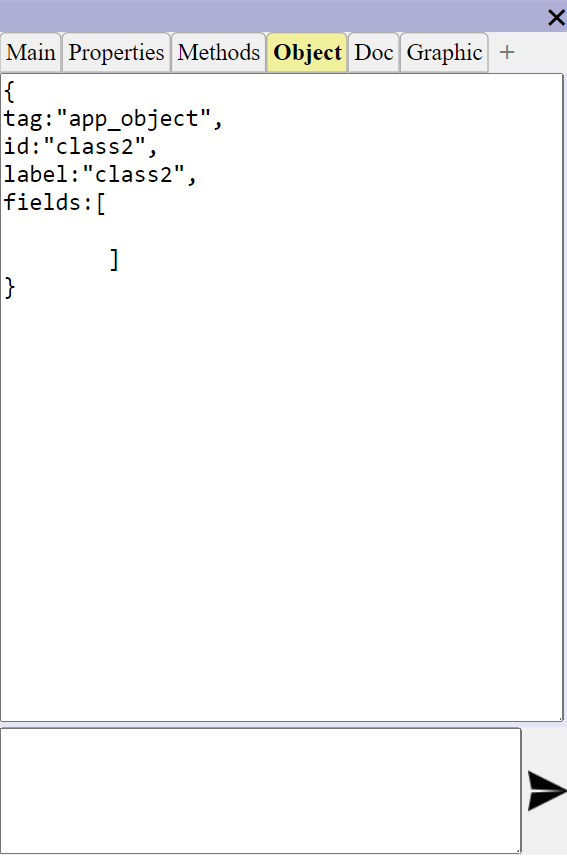
Il est possible de définir un objet de la même manière qu’on ferait un registerObject(). Par défaut il n’y en a aucun :



On utilise la commande object

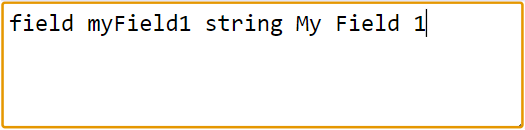


Une définition d’objet apparaît :



#### Ajout de champs

On utilise la commande ***field***. Elle est identique à celle utilisée pour les *form* et les *grid* :



La définition du nouveau champ apparaît :

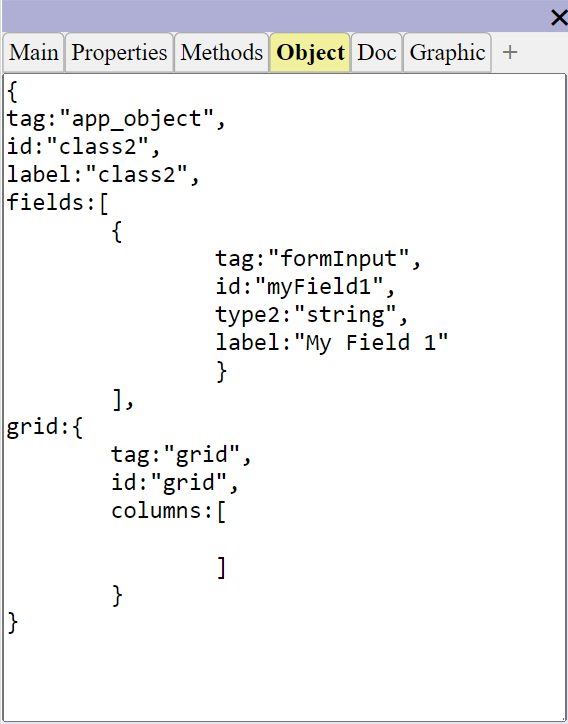


#### Ajout d’une grid

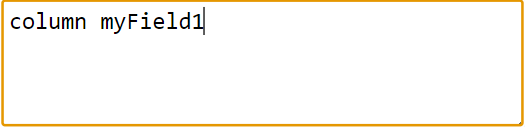
On utilise la commande grid :



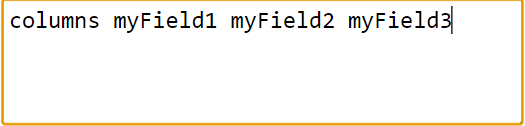
La définition de la grid apparaît :



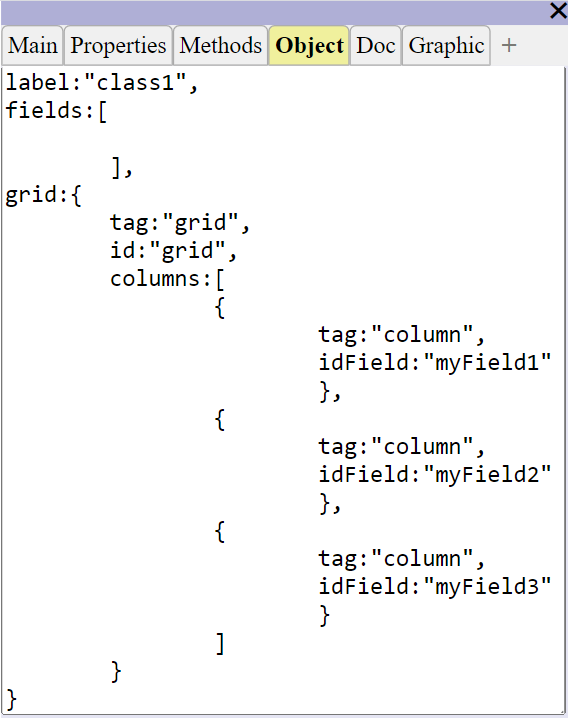
Pour lui ajouter une colonne on utilise la commande column



Ou la commande columns :



Les définitions de colonnes apparaissent :



#### Ajout d’actions

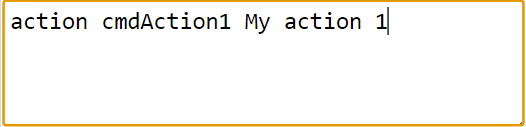
On utilise la commande actions



Cela ajoute le tableau actions :



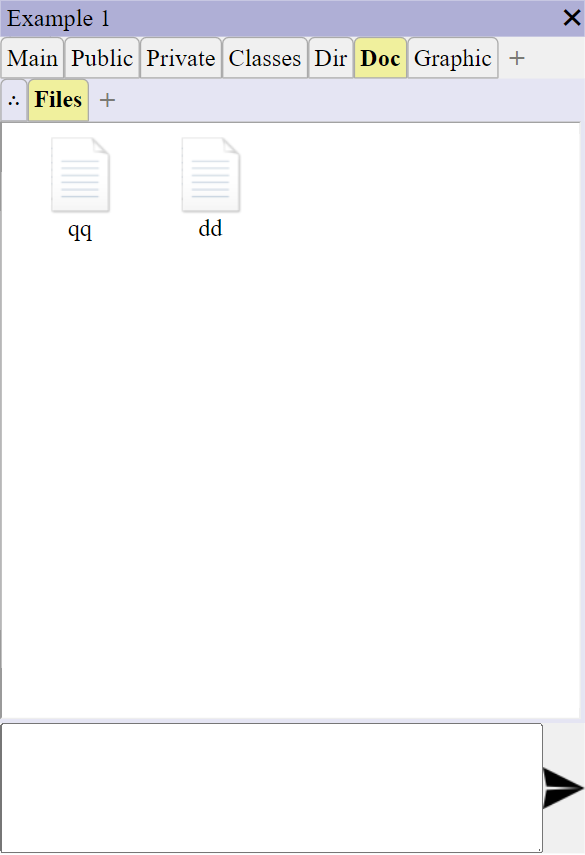
On utilise la commande action pour ajouter une nouvelle action :



Si le tableau des actions n’existe pas encore celui-ci est automatiquement créé.

### Gestion de fichiers

Toute application, classe et élément HTML comprend un onglet ***Doc*** comprenant à son tour un sous-onglet ***Files*** :



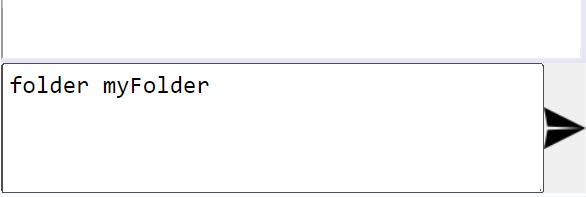
#### Création d’un fichier

Commande ***file***



#### Création d’un dossier

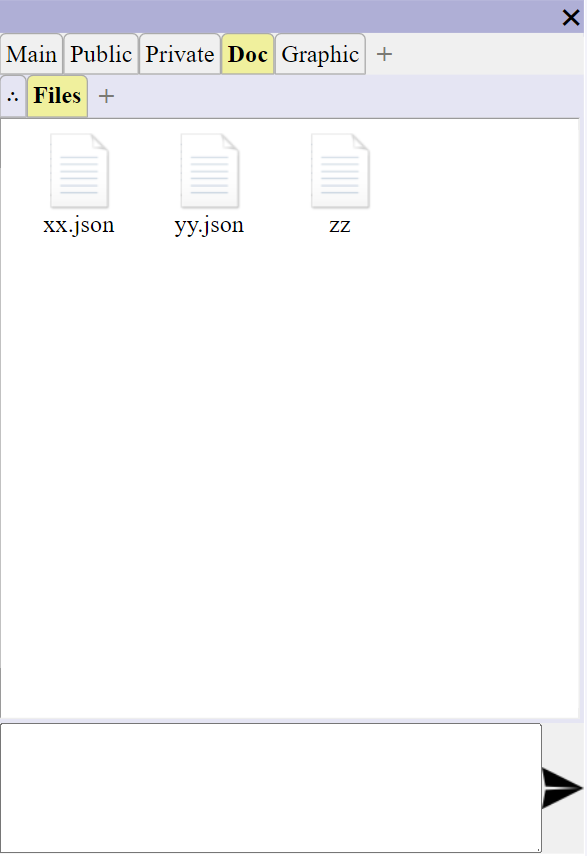
Commande ***folder*** (ou ***fdl***)



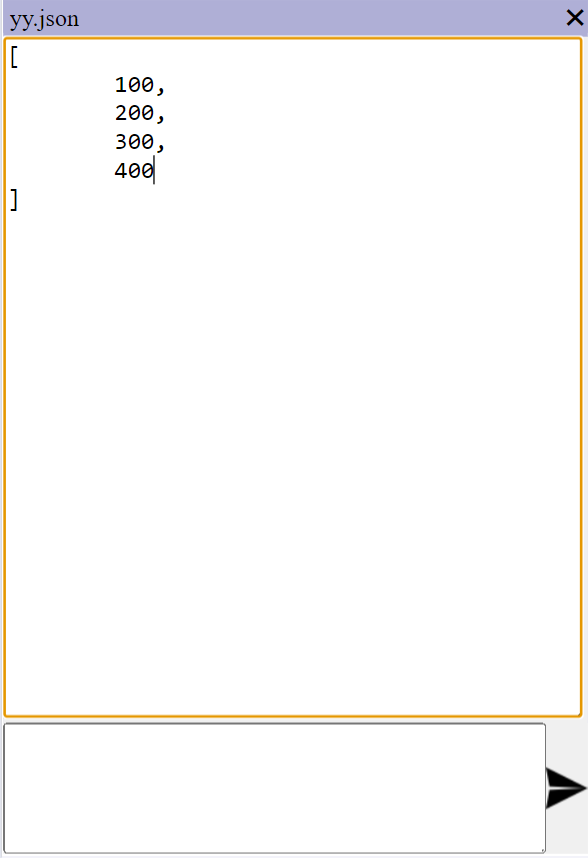
Bien sûr un dossier peut à son tour comprendre d’autres dossiers et des fichiers quelconques

#### Ouverture d’un fichier

Il suffit de cliquer sur un fichier pour l’ouvrir :



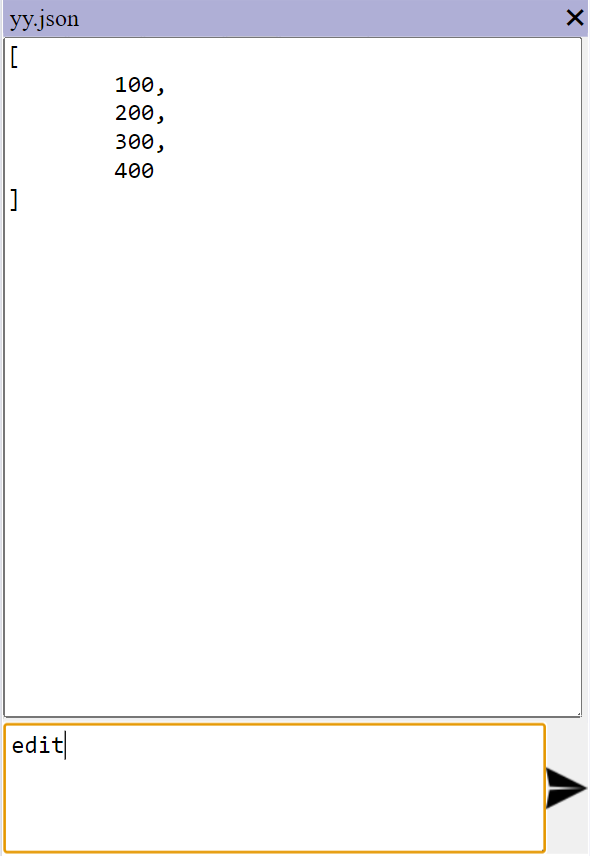
Le nom du fichier apparaît dans la barre de titre :



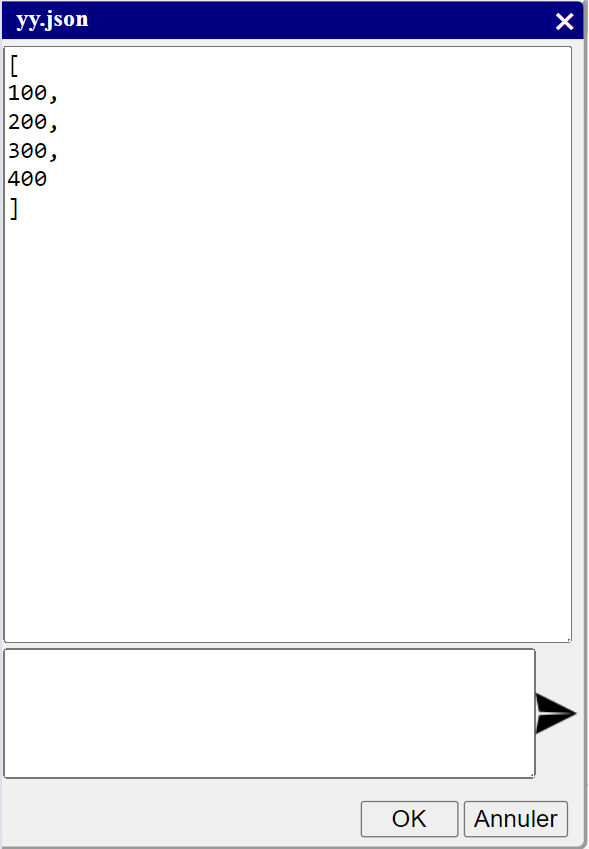
#### Edition d’un fichier

Le fichier étant ouvert,

* Taper la commande ***edit***



Si le fichier peut être parsé en tant que JSON il s’ouvre soit en tant qu’objet soit en tant que tableau (-> voir *Objets* et *Tableaux*)

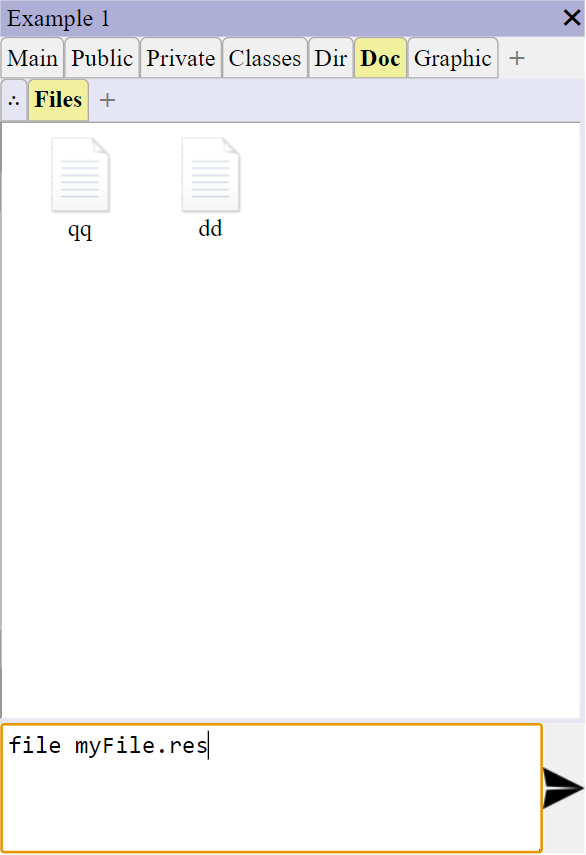


On peut alors éditer son contenu en écrivant directement dans l’éditeur avec les commandes *array*, *obj*, *del*.

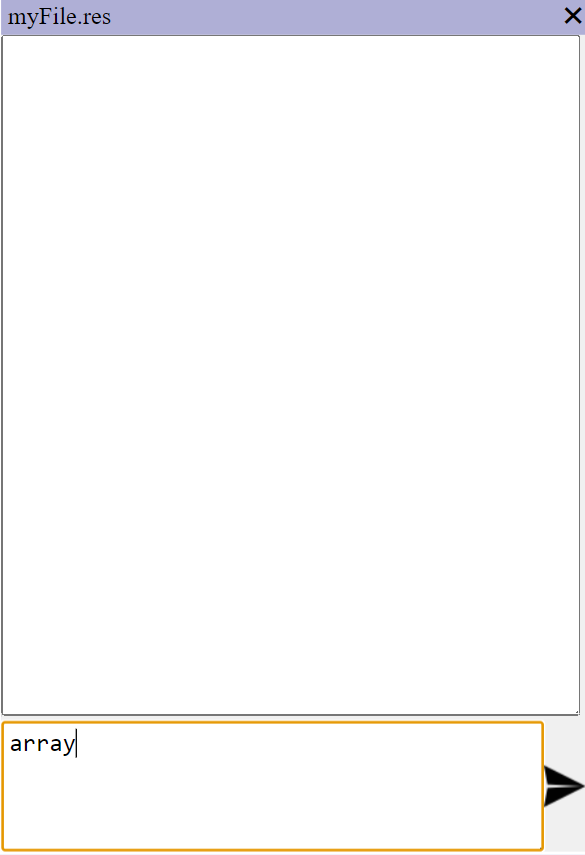
#### Créer un fichier json

Dans un onglet *Doc*

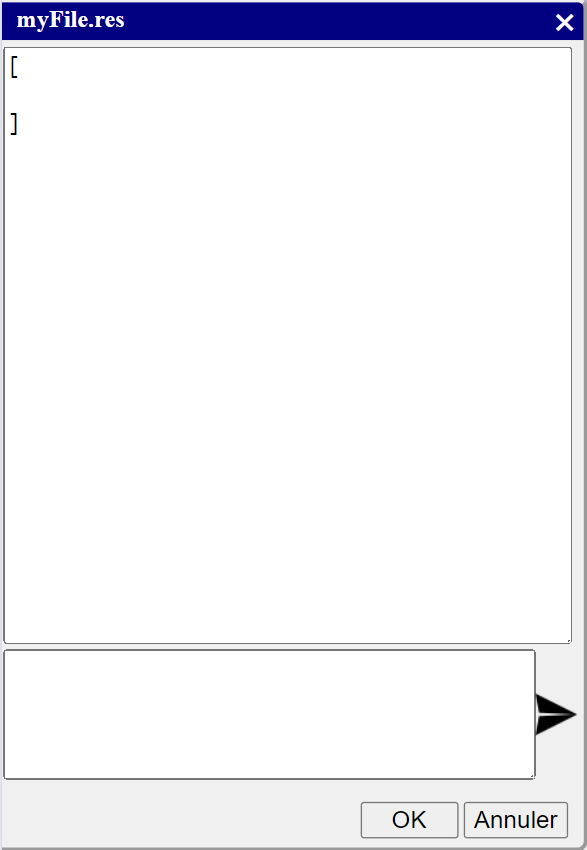
* Créer un fichier quelconque (peu importe l’extension éventuelle) :



* L’ouvrir
* Taper ***array*** (pour un tableau) ou ***obj*** (pour un objet) :



Le fichier est prêt à être édité :



* Cliquer sur ***OK*** à la fin (Attention, cliquer sur la croix équivaut à annuler)

#### Renommer un fichier

On utilise la commande ***rename*** :

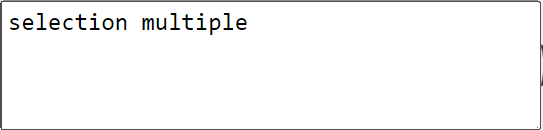


## Sélection

On active le mode **sélection** par la commande ***selection*** (ou ***sel*** ou ***s***)



On peut activer la sélection multiple par le paramètre ***multiple*** (ou ***mult*** ou ***m***)



On peut basculer l’état **multiselection** par la commande ***m*** :



On peut désactiver la sélection par la commande ***unselect*** (ou ***unsel*** ou ***uns*** ou une nouvelle fois ***s***)



On peut sélectionner tout par la commande ***selectall*** (ou ***selall*** ou ***all***)



On désélectionner tout par la commande ***unselectall*** (ou ***unselall*** ou ***unall*** ou ***uall***) :



On peut sélectionner l’élément parent par la commande ***s*** :



On peut ensuite ouvrir cet élément par la commande ***open*** :



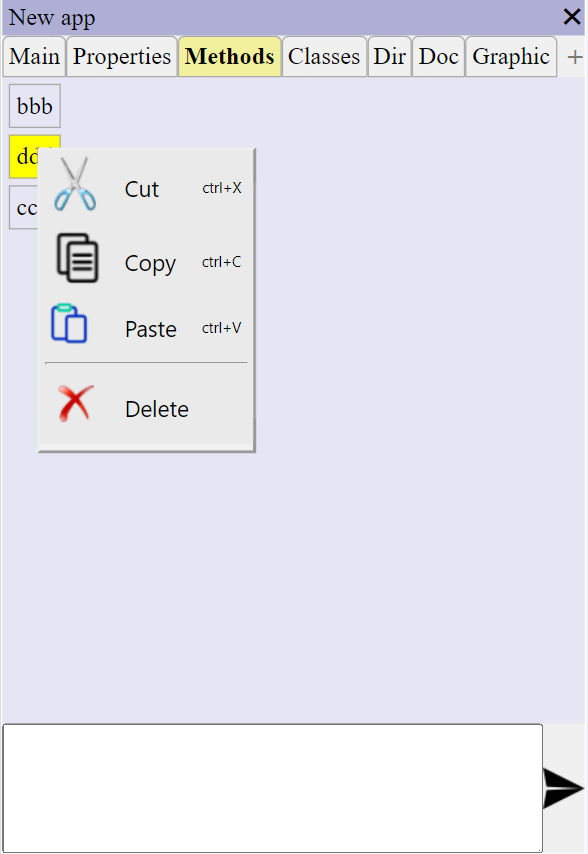
## Copier-coller

Il est possible d’effectuer des couper / copier – coller sur :

* Les propriétés
* Les méthodes
* Les classes
* Les documents
* Les éléments HTML

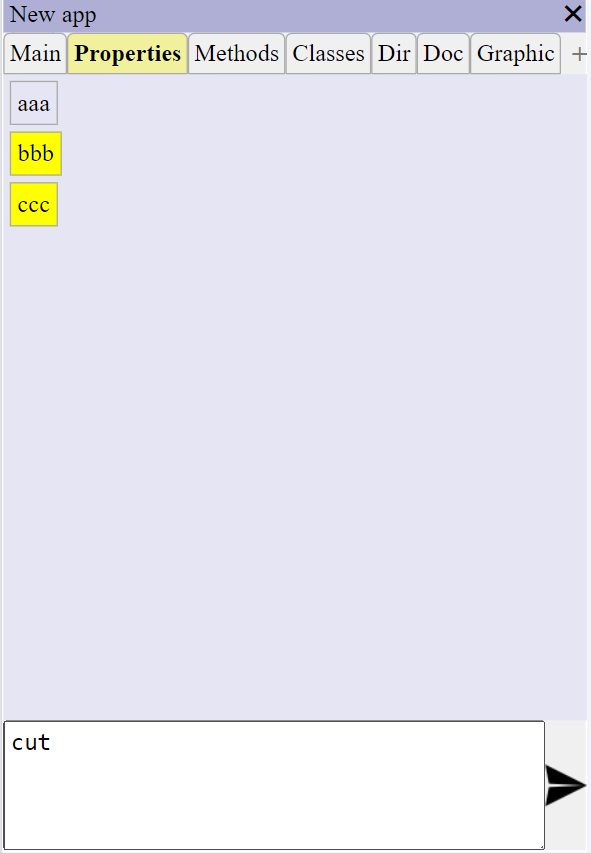
Pour cela,

* Passer en mode sélection
* Cliquer sur le ou les éléments pour les ajouter à la sélection
* Clic long sur un élement > ***Couper*** (ou ***Copier***)



**Attention**

Ce menu contextuel ne concerne que l’élément sur lequel on fait le clic droit. En cas de sélection multiple il faut utiliser à la place la commande ***cut*** ou la commande ***copy*** :



Pour coller,

* Clic droit sur la partie vide > ***Coller*** :



## Eléments HTML

### Généraux

* ***id*** *monId* : affecte l’id *monId*



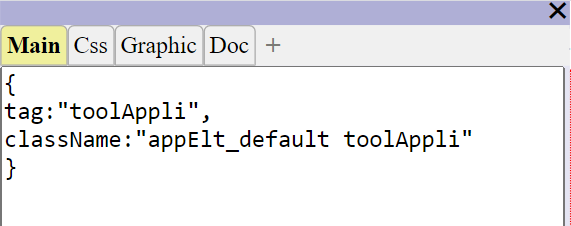
* ***className*** (ou ***cls***) *maClasse* : ajoute la classe *maClasse*



* cls1

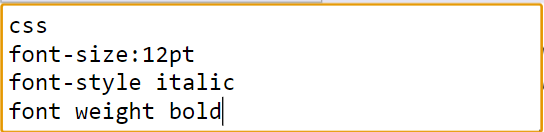


Elle ajoute une classe dont le nom est le même que le tag :



Cela permet de résoudre des problèmes d’affichage dans l’application car la classe *appElt\_default* peut écraser la classe par défaut de l’élément.

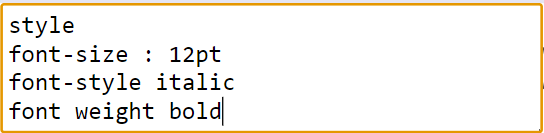
* ***css***



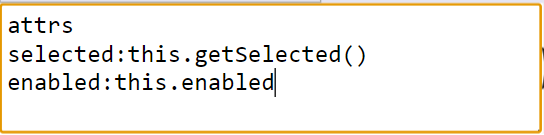
Les deux points (:) ne sont pas obligatoire (sauf en cas d’ambigüité) : ils se placeront par défaut au niveau du dernier espace trouvé.

Les tirets ne sont pas obligatoires avant les deux points (le nom de la propriété). Ils sont par contre nécessaires pour la valeur.

* ***styles*** (ou ***style***)



* ***attributes*** (ou ***attrs*** ou ***attr***)



Le deux points (:) est facultatif : s’il est omis il sera considéré comme étant à la place du premier espace.

* ***isElt*** *myId*



* ***isObj*** *myId*



* ***dn*** *mainBox* (*Display Name*)

Affecte une fonction display à l’élément courant. Dans l’exemple suivant on veut lui affecter la fonction *displayMainBox*() :



Si le nom indiqué ne commence pas par display alors ce préfixe est ajouté automatiquement. Ainsi *dn mainBox* est équivalent à *dn displayMainBox*

### Police

* ***bold***
* ***italic***
* ***underline***

### Alignement

* ***flex***

equivalent à *display:flex*

* ***horz***

equivalent à *display:flex ; flex-direction:row;*

* ***vert***

equivalent à *display:flex ; flex-direction:column*;

* ***text-align*** (ou ***align***)

equivalent à *text-align :center*

* ***spacebetween*** (ou ***space***)

equivalent à *justify-content* : *space-between*

Permet d’écarter les éléments sur tout l’espace disponible :



### Marges

* **margin** *10*
* **padding** *10*

### bordures

* ***border*** *solid 1px red*

### Couleurs

* ***background*** (ou ***bg***) *red*

equivalent à *background-color*

* ***color*** (ou ***fg***) *red*

equivalent à *color*

Dimensions

* ***width*** *150*

equivalent à *width*

* ***height*** *50*

equivalent à *height*

### Contenu texte

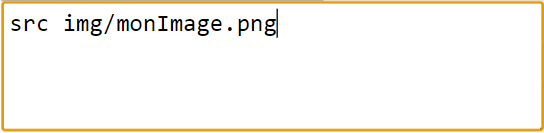
* ***html*** mon texte

### Retirer le style par défaut

* ***rs*** (=Remove Style)

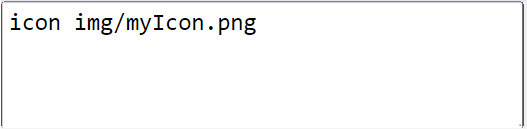
### Autres

* ***src***

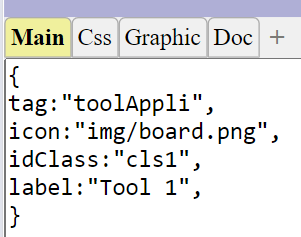


Le chemin et l’extension peuvent être omis (par défaut *img/* et *.png* respectivement)

* ***icon***



Elle renseigne l’éventuelle propriété icon de l’élement courant :



Là aussi le dossier par défaut et l’extension par défaut peuvent être omis.

### Créer des éléments enfants

#### Par scission de l’élement courant

* ***v*** : scinde verticalement en 3, l’élement central occupant le maximum d’espace (*flex:1*)

.classParent\_header  
displayHeader()

.classParent\_body  
displayBody()

.classParent\_footer  
displayFooter()

* ***h*** : même chose mais horizontalement

.classParent\_middle  
displayMiddle()

.classParent\_left  
displayLeft()

*.classParent\_right  
displayRight()*

* ***h2*** : scinde horizontalement en 2 parties égales

.classParent\_1

.classParent\_2

* ***h3*** : idem en 3 parties égales

.classParent\_3

.classParent\_2

.classParent\_1

* ***v2*** : scinde verticalement en 2 parties égales

.classParent\_1

.classParent\_2

* v3
* ***v3*** : idem en 3 parties égales

.classParent\_1

.classParent\_2

.classParent\_3&

* h 100 50

.classParent\_2

.classParent\_1

* h 100 50 50

.classParent\_1

.classParent\_2

* ***v*** 50px 50px 150px : idem avec affectation de tailles

.classParent\_1

.classParent\_2

.classParent\_3

* ***v*** 20% 80%

.classParent\_1

.classParent\_2

* ***v*** 50 50 150

myClass1

myClass2

myClass3

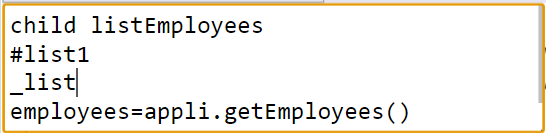
.classParent\_myClass1

.classParent\_myClass2

.classParent\_myClass3

#### Par simple ajout

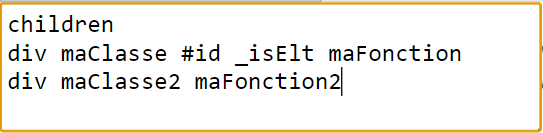
***addChild*** (ou ***child***)



Crée un élément avec :

* tag :*listEmployees*
* id :*list1*
* isObj :\_*list*
* employees=*appli.getEmployees*()

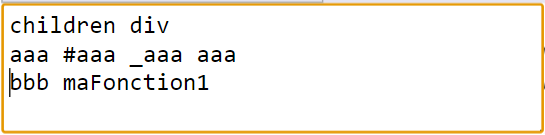
***addChildren*** (ou ***children***)



Crée 2 éléments enfants :

* un div de classe *maClasse*, avec id :*id*, isElt :\_*isElt* et de fonction d’affichage *displayMaFonction()*
* un div de classe *maClasse2* et de fonction d’affichage *displayMaFonction2()*

L’identifiant, le *\_isElt* et la fonction d’affichage ne sont pas obligatoires



Crée 2 élements de type *div*

##### div

* ***div***
* div *maClasse*
* div *maClasse1 maClasse2*
* div   
  *#div1*

Crée un div avec id=div1

* div

*\_div1*

Crée un div avec isElt=*\_div1*

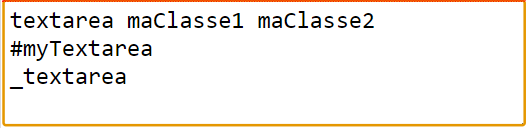
* div

*nomQuelconque*

Crée un div affiché par la fonction *displayNomQuelconque*()

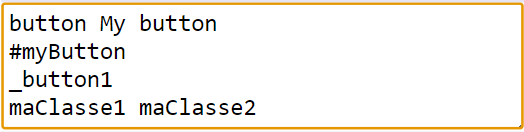
##### textarea

Commande ***textarea*** :

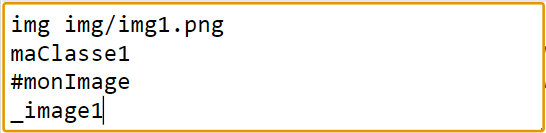


##### Bouton

Commande ***button*** :



##### img



Si *src* ne spécifie pas de chemin (pas de présence du caractère « / »), « *img/* » est automatiquement ajouté

Si l’extension n’est pas précisée, « .*png* » est automatiquement ajouté

Exemple :



##### tabs



##### tab



**Attention**

Il faut obligatoirement créer ensuite un élément div à l’intérieur par la commande *div*



##### Element quelconque

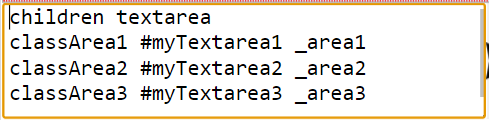
On utilise la commande **addchild** (ou ***child*** ou ***elt***) :



##### Ajouter plusieurs éléments

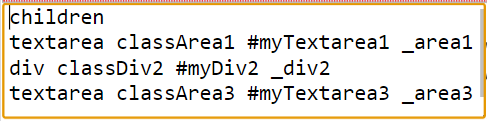
On utilise la commande addchildren (ou children). 2 cas :

* Si tous les éléments doivent être de même tag :



Pour chaque ligne le premier argument correspond toujours à la **classe**, pour les suivants cela dépend du comment ils commencent (***#*** pour *id*, ***\_*** pour \_*isElt* ou \_*isObj*)

* S’ils sont de tags différents :



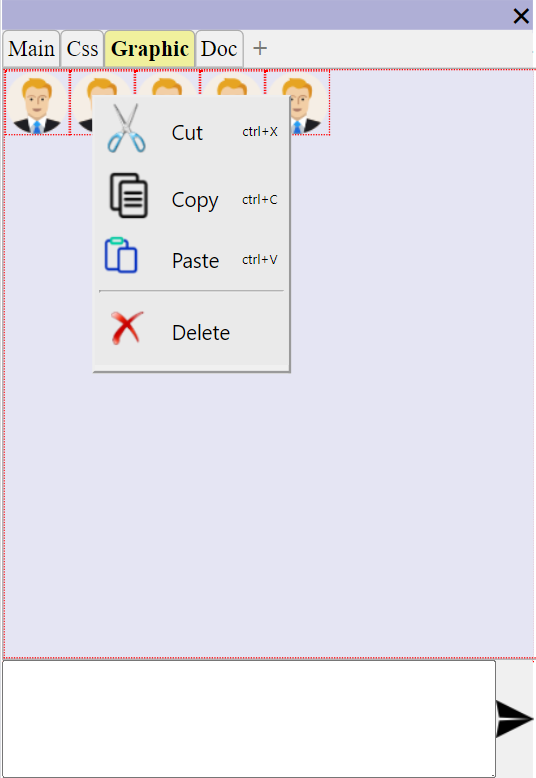
Cette fois chaque ligne commence par le tag suivi de la classe et un nombre variable de paramètres

### Supprimer des éléments

#### 1 seul élement

On peut utiliser l’option ***Supprimer*** dans le menu contextuel :

* Passer en mode *sélection*
* Clic droit sur l’élément > ***Supprimer*** :



#### Plusieurs éléments

En cas de sélection multiple on utilise la commande ***delete*** :

* Passer en mode sélection
* Sélectionner le (ou les) élement(s)
* Taper ***delete***



**Attention**

A ne pas confondre avec la commande *del* qui est utilisée spécifiquement aux propriétés et méthodes (mais on peut aussi utilise la commande *delete* pour ces dernières).

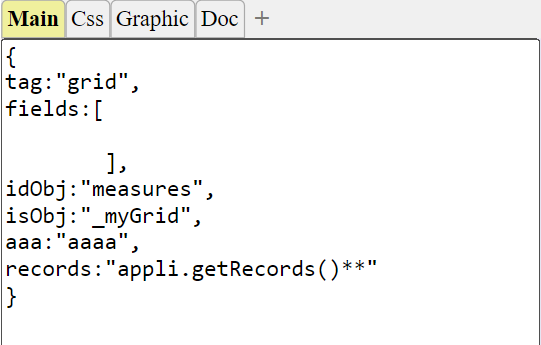
#### Tous les éléments enfants

Pour supprimer tous les éléments enfant de l’élément courant on utilise la commande ***clear*** :



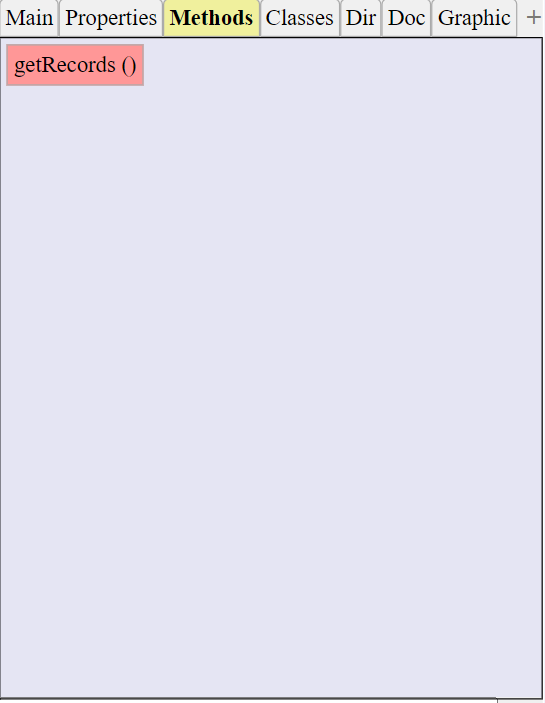
### Propriété faisant appel à une fonction

Dans l’exemple suivant la propriété records fait appel à la fonction appli.getRecords() :

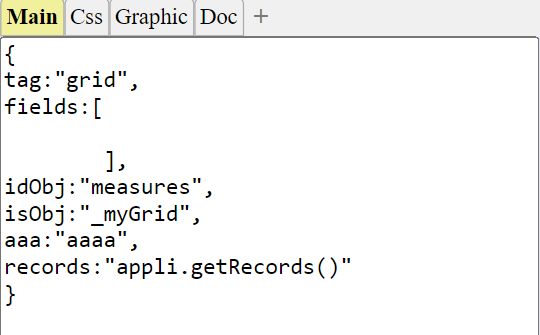


On doit placer cet appel entre guillemets afin d’éviter les problèmes de parsing.

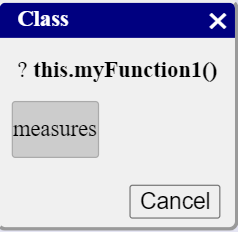
La double astérisque (\*\*) permet d’indiquer au système qu’il faut créer une nouvelle définition (ici au niveau de l’application).



Celui-ci la retirera ensuite automatiquement :

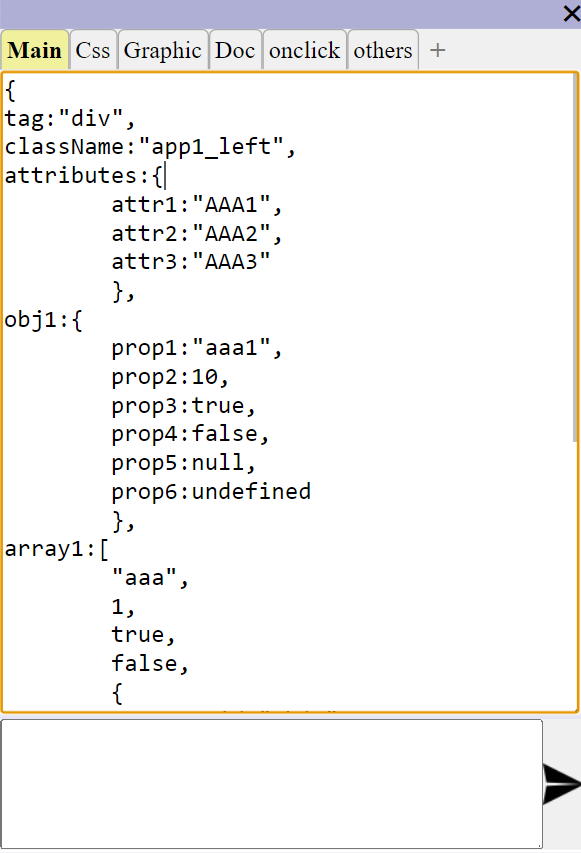


Si la référence indiquée n’appartient pas à *appli* mais à ***this***, une boîte de dialogue apparaîtra automatiquement demandant à quelle classe doit appartenir la future méthode :



### Objets

Dans l’exemple ci-dessous l’une des propriétés d’un div est un objet :

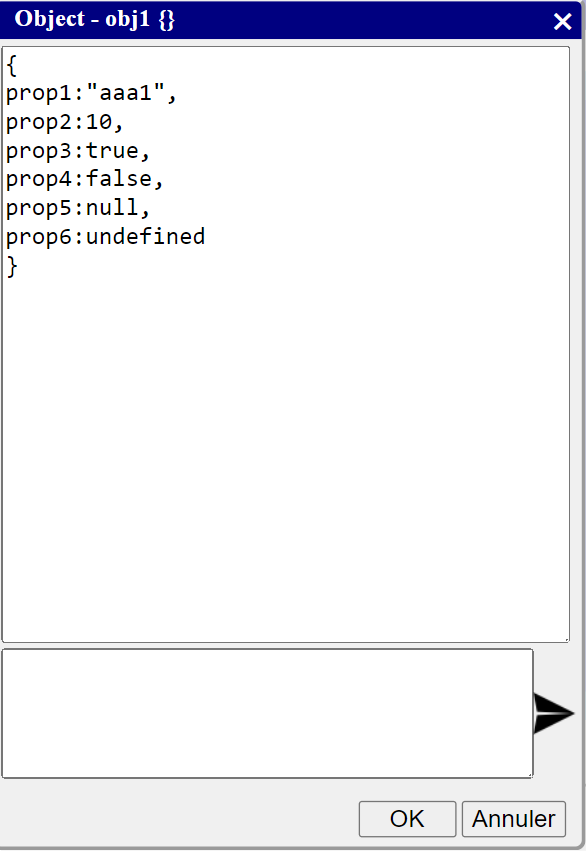


#### Edition d’une propriété de type objet

On utilise la commande ***obj*** suivie du nom de la propriété



Un **éditeur d’objet** apparaît :



#### Ajout d’une nouvelle propriété de type objet

On utilise la même commande cette fois en indiquant le nom de la future propriété :



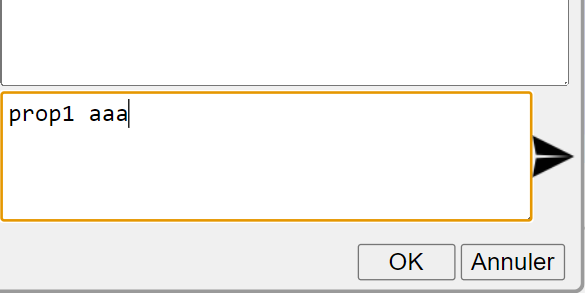
On accède à l’éditeur d’objet vu précédemment

#### Edition de l’objet

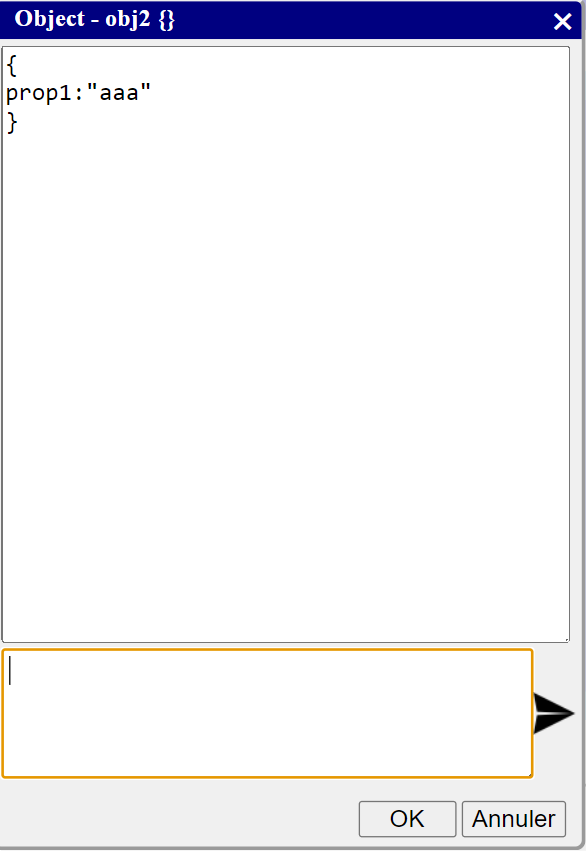
##### Affectation d’une propriété scalaire

Depuis l’éditeur d’objet, il suffit de taper les propriétés sous la forme :

nomPropriété valeurPropriété



L’objet est alors mis à jour :



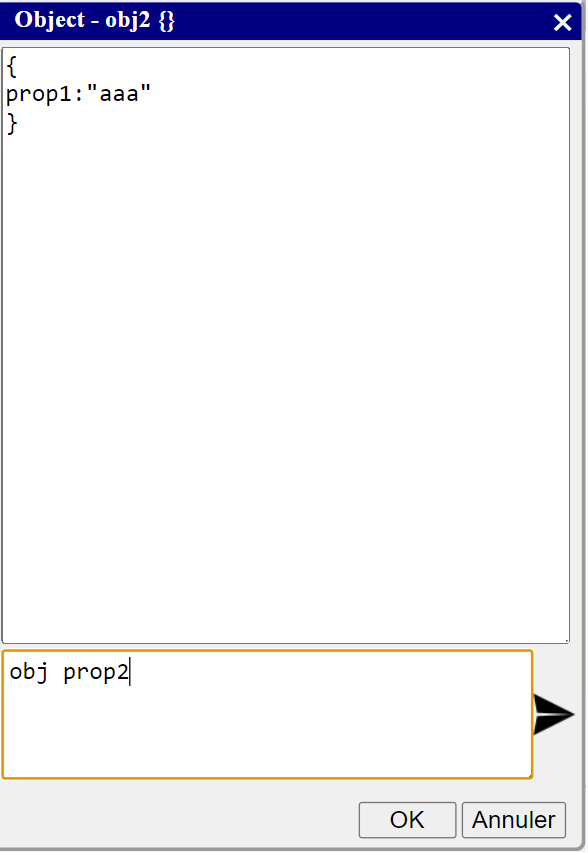
**Remarque**

Il est inutile d’indiquer des guillemets pour les valeurs alphanumériques, cela est détecté automatiquement. De même sont reconnues les valeurs *true*, *false*, *null* et *undefined*

##### Affectation de propriété de type objet

On peut ajouter une propriété qui est elle-même un objet.

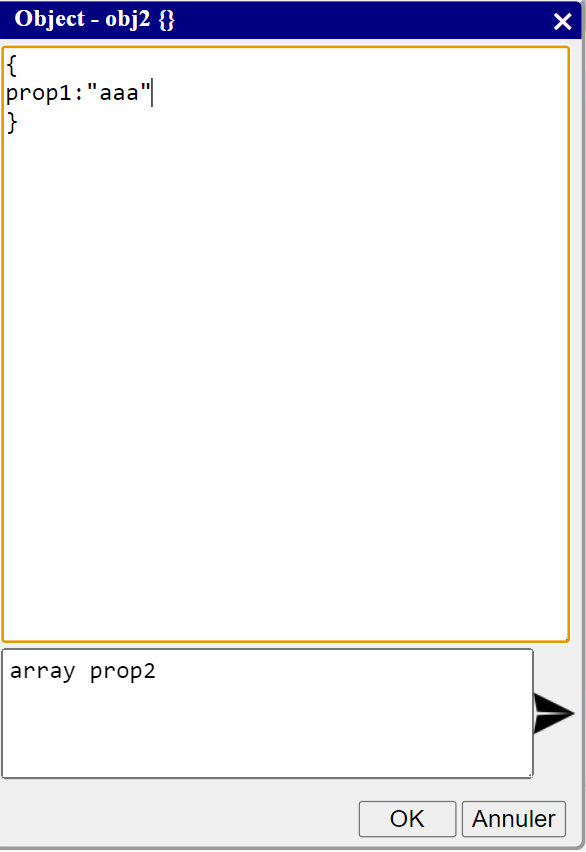
On utilise pour cela la commande ***obj*** à la laquelle on indique le nom de la future propriété :



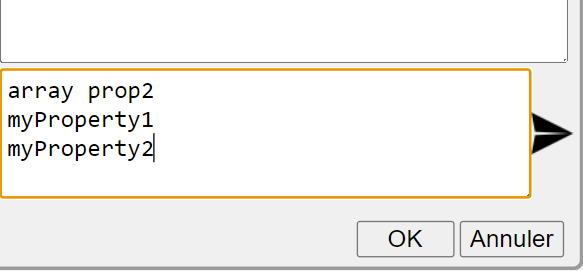
Cela ouvre un nouvel éditeur d’objet auquel on peut ajouter des propriétés.

##### Affection de propriété de type tableau

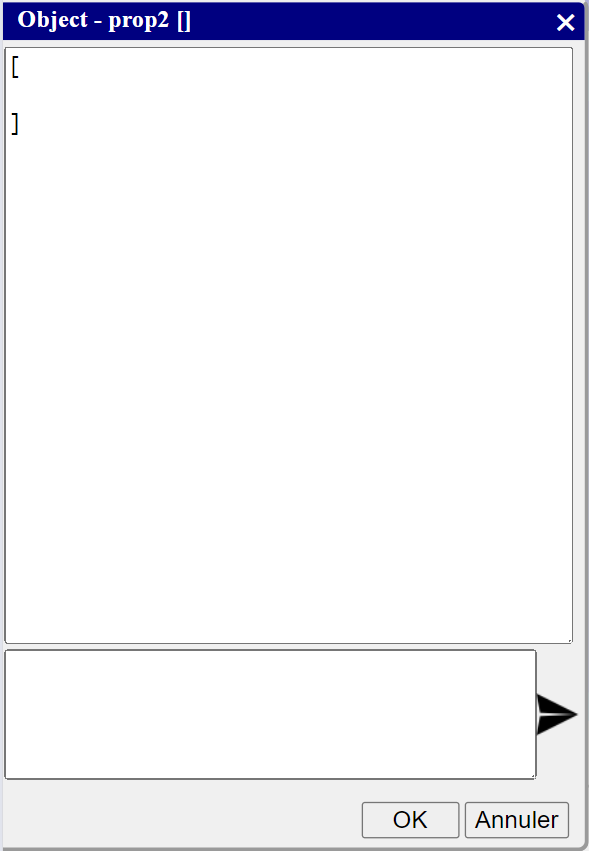
On utilise pour cela la commande ***array*** à la laquelle on indique le nom de la future propriété



On peut aussi ajouter la liste des propriétés attendues par les futurs enregistrements :



Cela ouvre l’**éditeur de tableau** :



* Voir *Tableaux*.

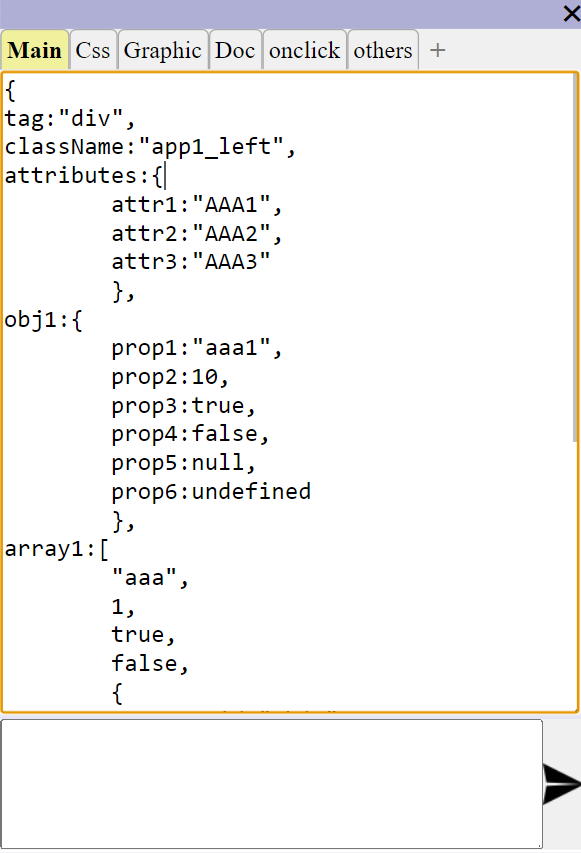
#### Supprimer une propriété

Commande ***del***



### Tableaux

Dans l’exemple ci-dessous l’une des propriétés d’un div est un tableau :

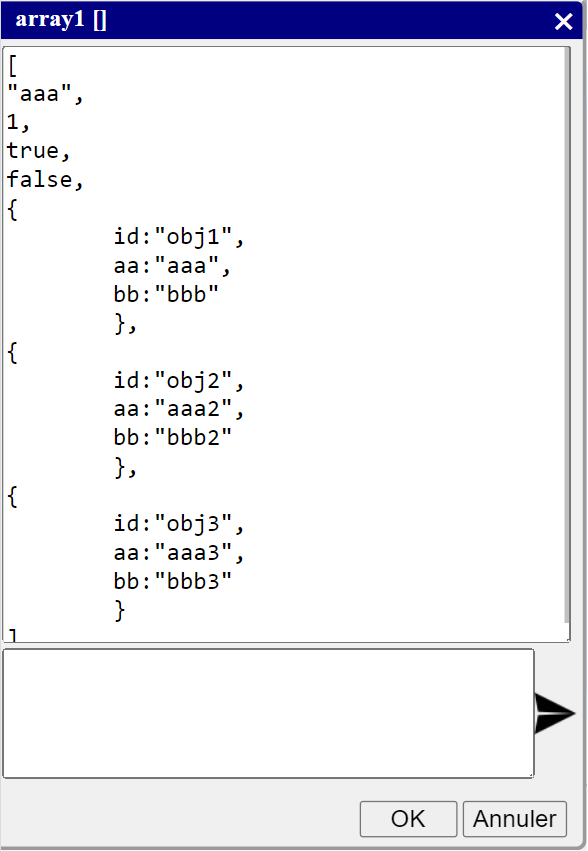


#### Edition d’une propriété de type tableau

On utilise la commande ***array*** suivie du nom de la propriété



L’**éditeur de tableau** apparaît :

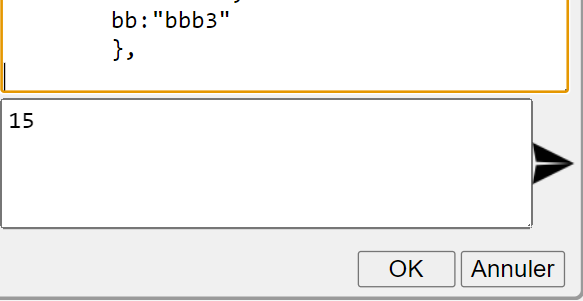


On peut y modifier directement le tableau (et les objets ou tableaux inclus).

On referme cet éditeur en cliquant sur OK.

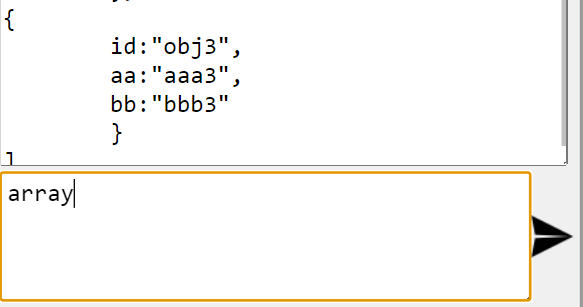
##### Ajout d’un enregistrement scalaire

On peut taper directement la valeur :



##### Ajout d’un enregistrement de type tableau

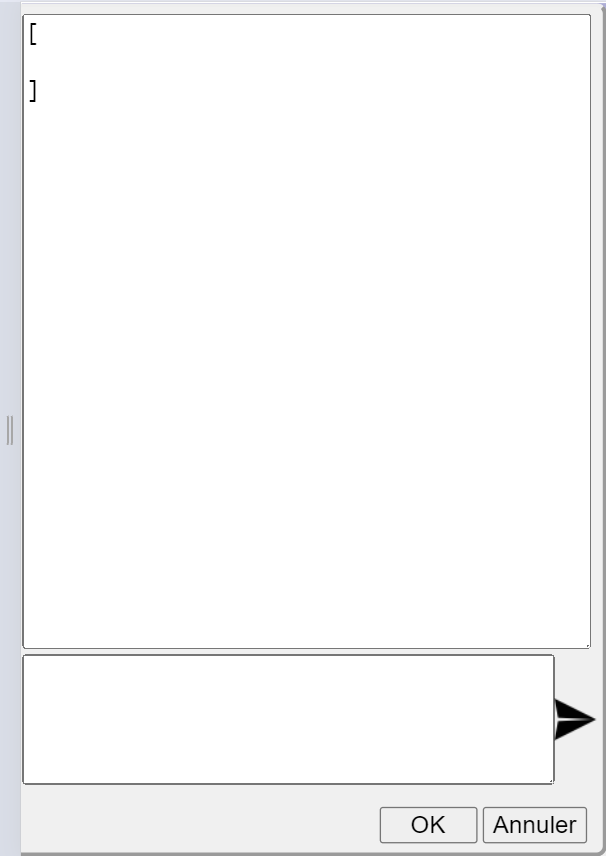
Pour insérer un tableau dans un tableau on utilise la commande ***array*** :



On peut aussi indiquer en argument la liste des propriétés des objets qu’on compte insérer dans ce deuxième tableau :



Un autre éditeur de tableau apparaît :



##### Ajout d’un enregistrement de type objet

On utilise la commande ***obj*** :



obj  
prop1:aaa  
prop2:2  
prop3:null  
prop4:undefined  
prop5:-1  
prop6:0  
prop7:array  
prop8:obj

On peut remplacer les deux-points (:) par des espaces

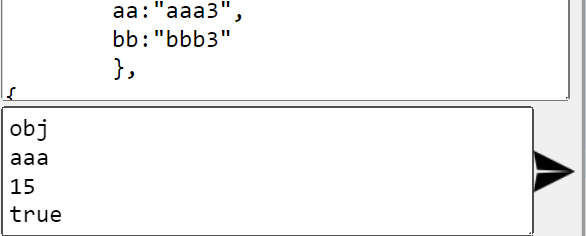
obj  
prop1 aaa  
prop2 2  
On peut indiquer des tableaux ou des objets inclus à leur tour par les mots-clés ***array*** et ***obj***[Pas implémenté pour le moment]

obj  
prop7:array  
prop8:obj

Si on doit ajouter plusieurs enregistrements à la suite on peut indiquer la liste des propriétés dès l’ouverture du tableau :



On n’aura plus qu’à indiquer les valeurs pour chaque enregistrement sans avoir à rappeler le nom de la propriété :



##### Supprimer un enregistrement

On utilise la commande ***del***

###### Supprimer le 1er enregistrement



###### Supprimer le dernier enregistrement



###### Par numéro d’index

Supprime l’enregistrement numéro indexEnregistement (à partir de 0).



###### Supprimer tous les enregistrements tels que propriété = valeur

Supprime tous les enregistrements tels que id=user1



###### Supprimer tous les enregistrements

***del all*** (ou ***clear***)



## Eléments JSA

### Form2

#### Création

On utilise la commande ***form2*** (ou ***form***) :

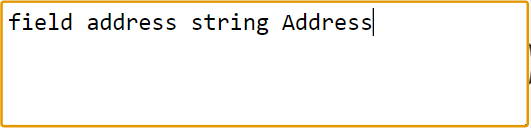


form2  
#myForm  
\_form  
firstName string First name  
lastName string Last name  
dateBirth date Date birth

Pour chaque champ (field) seul l’identifiant est obligatoire. Le type sera string par défaut.

#### Ajouter un champ

On utilise la commande ***field*** :



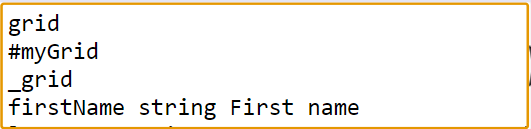
### Grid

#### Création

On utilise la commande ***grid***.

Plusieurs façons de l’utiliser :

* En indiquant les fields (comme pour les *form2*) :



grid

#myGrid

\_grid

firstName string First name

lastName string Last name

dateBirth date Date birth

* En indiquant la définition d’un objet (par son identifiant). Il faut bien sûr que cet objet ait été définit par la fonction appli.registerObject()

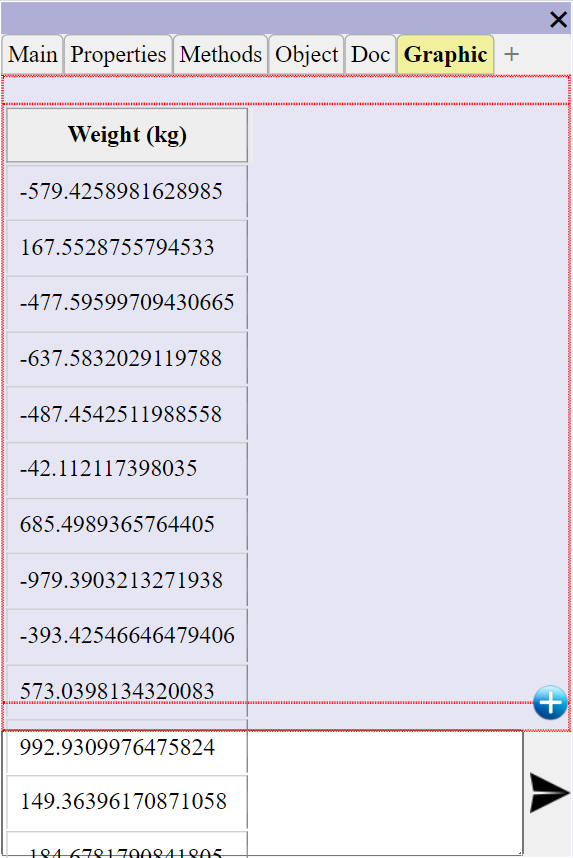


form2  
#myForm  
\_form  
firstName string First name  
lastName string Last name  
dateBirth date Date birth

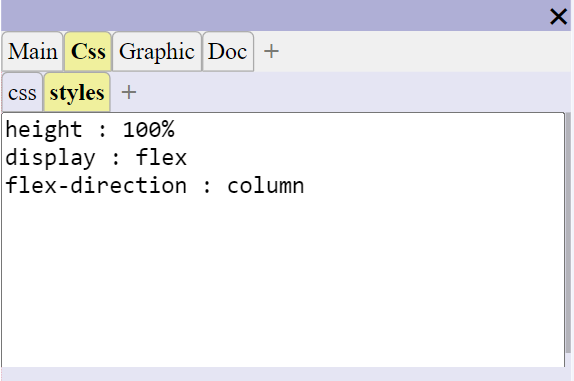
Pour chaque champ (field) seul l’identifiant est obligatoire. Le type sera string par défaut.

**Remarque**

Il se peut que la grid déborde à un moment de son élement parent :

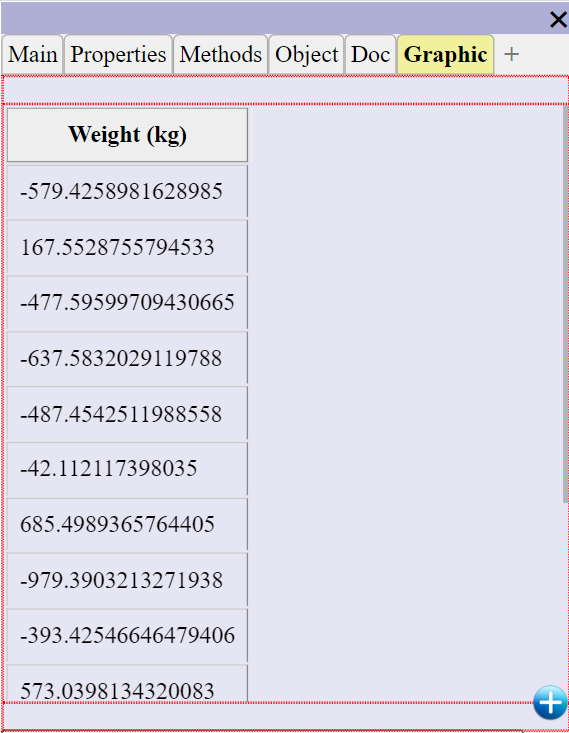


Cela peut être résolu en lui ajoutant les styles suivants :



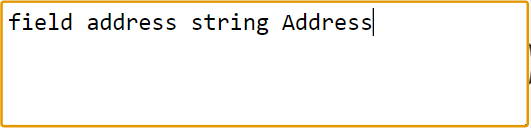
height : 100%  
display : flex  
flex-direction : columns

Un ascenseur apparait désormais quand nécessaire



#### Ajouter un champ

On utilise la commande ***field*** :



#### Remplissage automatique

On utilise la commande ***filldata*** (ou ***fill*** ou ***data***)



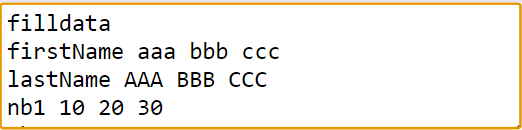
Par défaut il crée 15 lignes. On peut changer cette valeur :



Par défaut il remplit tous les champs :

* Numériques avec des *0*
* Alphanumériques avec des *#*
* Dates par *01/01/2001*

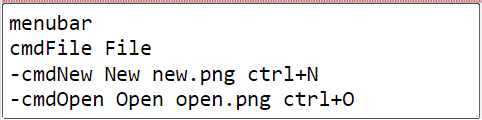
On peut préciser les valeurs possibles



filldata 100  
firstName aaa bbb ccc  
lastName AAA BBB CCC  
nb1 10 20 30  
nb2 100-500  
dateBirth 01/01/1950-31/12/2020

### Menubar

On utilise la commande ***menubar*** (ou ***mb***):



menubar  
#myMenubar  
\_menubar  
cmdFile File  
-cmdNew New new.png ctrl+N  
-cmdOpen Open open.png ctrl+O  
-\_\_\_  
-cmdSave Save save.png ctrl+S  
-\_\_\_  
-cmdQuit Quit  
cmdEdit Edit  
-cmdCut Cut cut.png  
-cmdCopy Copy copy.png Ctrl+C  
-cmdPaste Paste paste.png Ctrl+P  
-cmdCopyTo CopyTo  
--cmdCopyLeft ToTheLeft  
--cmdCopyBottom ToTheBottom  
cmdInsert  
-cmdInsertImg Image  
-cmdInsertText Text

Seuls l’identifiant et le label sont obligatoires

Noter qu’il est impossible de créer des espaces dans les labels. Il faudra les rajouter plus tard directement dans l’onglet *Main*.

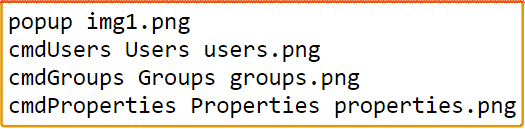
Si on veut indiquer un raccourci clavier sans avoir d’image il faut indiquer null à la place de l’image

cmdAction Action null Ctrl+B

### Popup

On utilise la commande ***popup*** :

#### Pour une image

  
popup menu2.png   
#myPopup  
\_myPopup  
cmdUsers Users users.png  
cmdGroups Groups groups.png  
cmdProperties Properties properties.png

#### Pour un bouton

popup My button  
#myPopup  
\_myPopup  
cmdUsers Users users.png  
cmdGroups Groups groups.png  
cmdProperties Properties properties.png

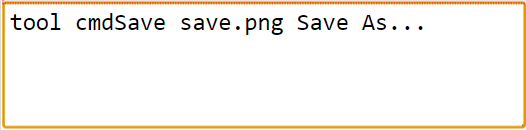
### Toolbar

On utilise la commande ***toolbar*** (ou ***tb***)

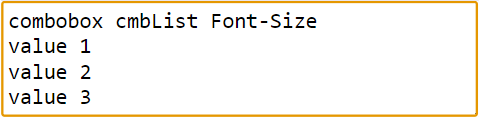


Puis on y rajoute des outils au fur et à mesure par les commandes :

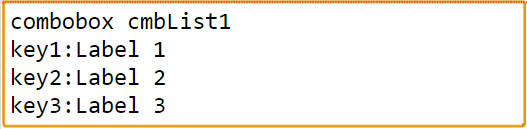
* ***tool*** :



* ***combobox*** (ou ***combo***):



Ou par paires clés-valeurs :



* ***toolappli*** (ou ***toolapp***, ***applitool*** ou encore ***apptool***) :



Elle crée des éléments *toolAppli* :

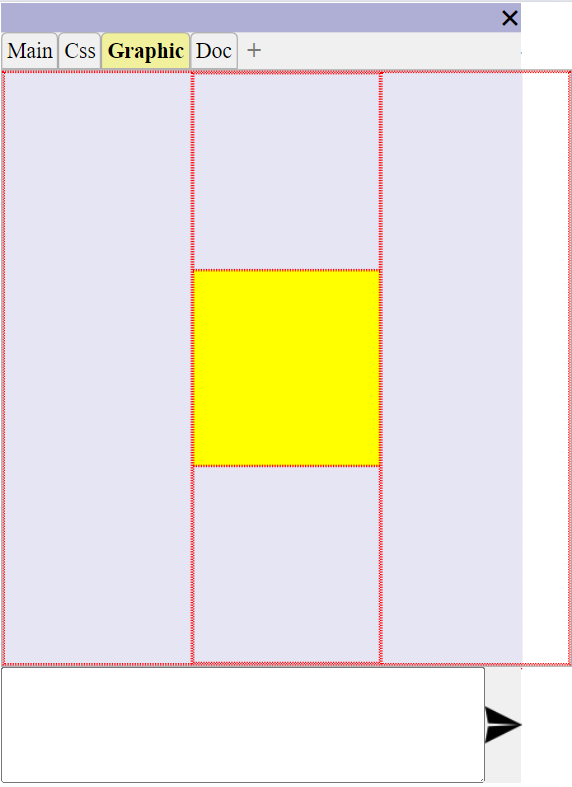


{  
tag:"toolAppli",  
icon:"img/board.png",  
idObj:"cls1",  
label:"Tool 1"  
}

Pour rappel un *toolAppli* est un bouton destiné à ouvrir un objet *screen* ou dlg : ce dernier est indiqué par la propriété *idObj* (il doit avoir été créé préalablement et être accessible depuis l’objet *appli* : *appli[idObj]*. Ce comportement par défaut peut être changé en réimplémentant sa méthode *onclick*().

## Déplacement des éléments

Il faut d’abord sélectionner l’élement à déplacer :



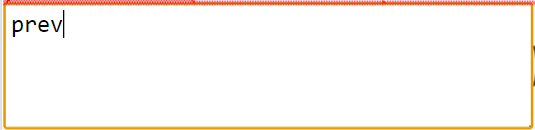
### Commande générale

Commande **move**



### Monter

Commande ***prev***



### Descendre

Commande ***down***

### Monter en premier

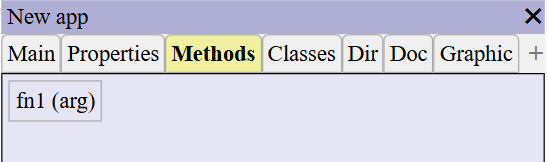
Commande ***first***

### Descendre en dernier

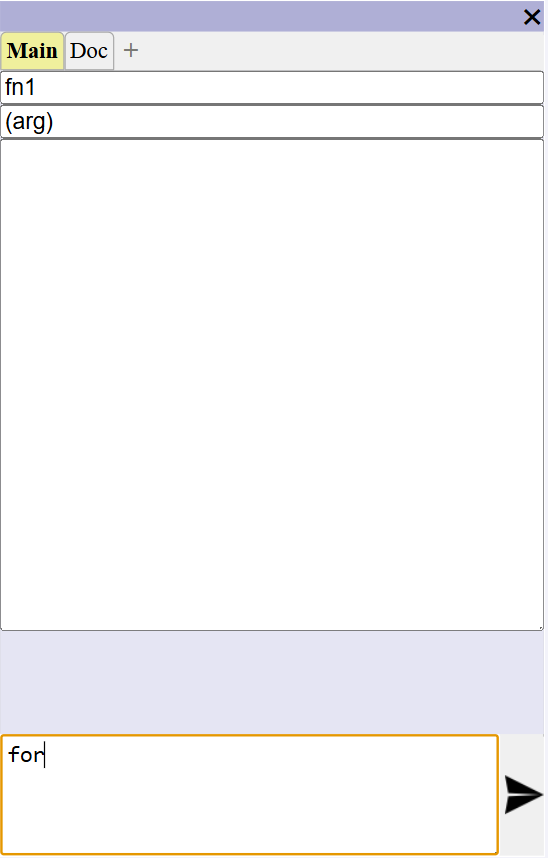
Commande ***last***

# Structures de contrôle

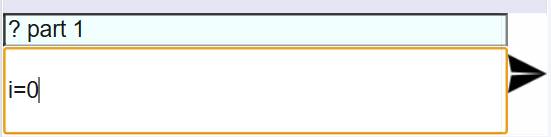
Elles sont utilisables dans les fonctions :



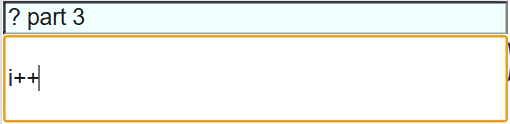
## for



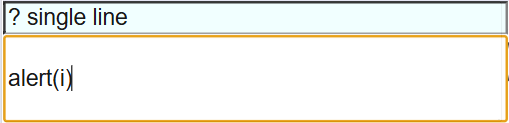
Les 3 parties de ce qui se trouvera entre parenthèses sont demandées successivement :



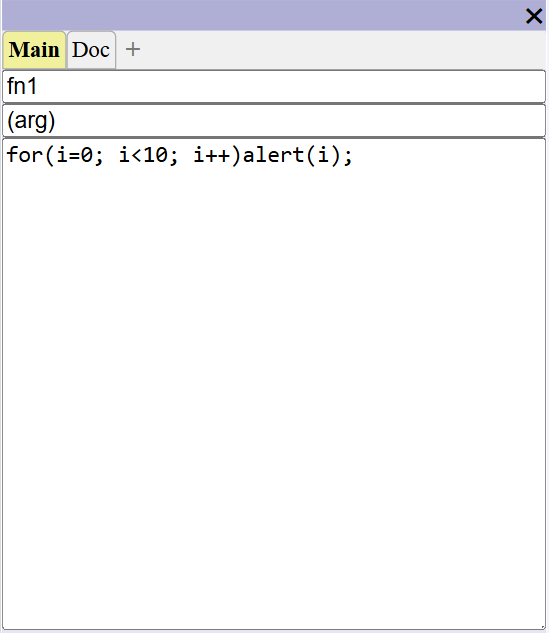




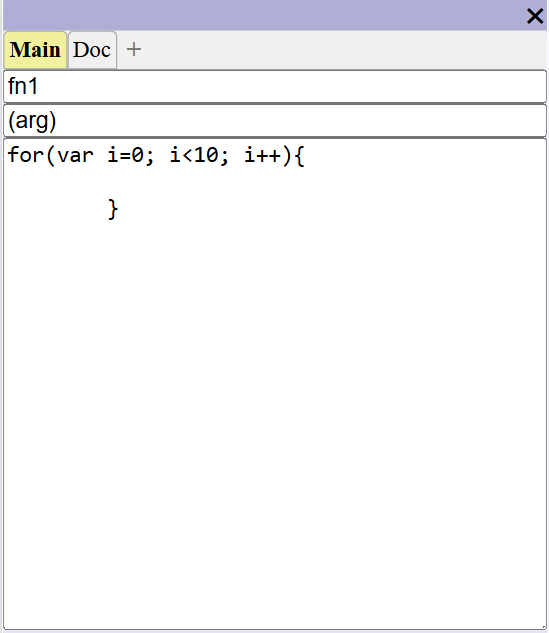
Puis une ligne de code est demandée :



Si elle est renseignée elle apparaîtra en ligne :



Sinon ce seront des accolades vides :



## foreach



## map



## while



## if



## switch



## promise



## dlg

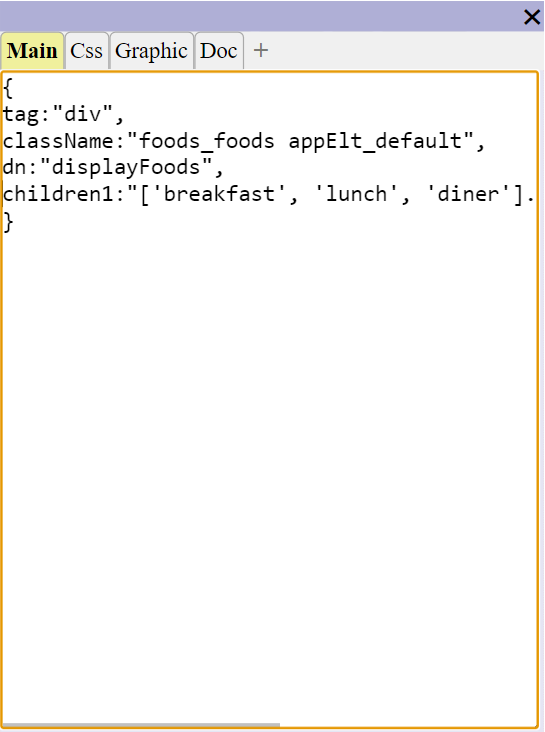


## openpopupmenu (openpopup, openmenu, openmenupopup)



# Divers

Utilisation de la propriété d’élément ***children1***



# Documentation

doc

keywords (words, word, key)