

## Créer une application avec le générateur d'applications

### Objectif :

Créer une application

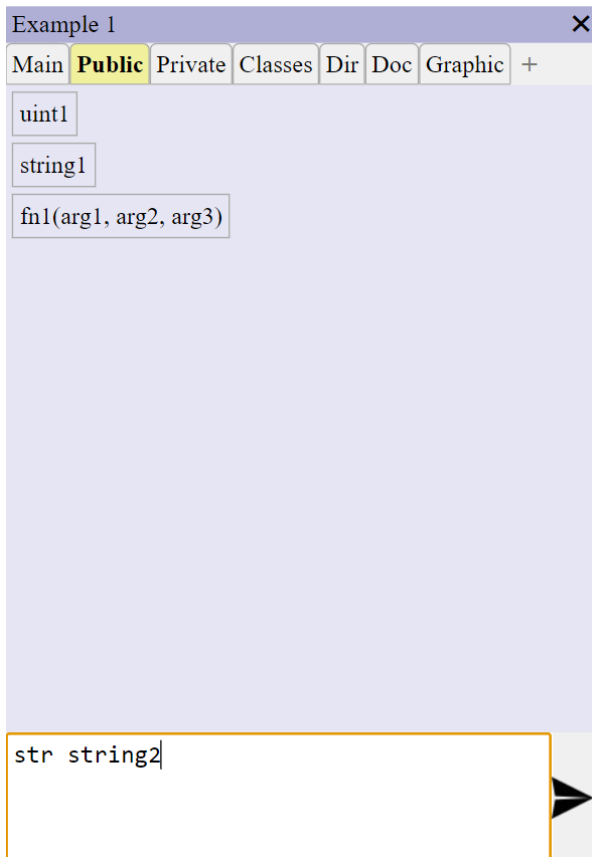
### Mise en place :

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 [{}]

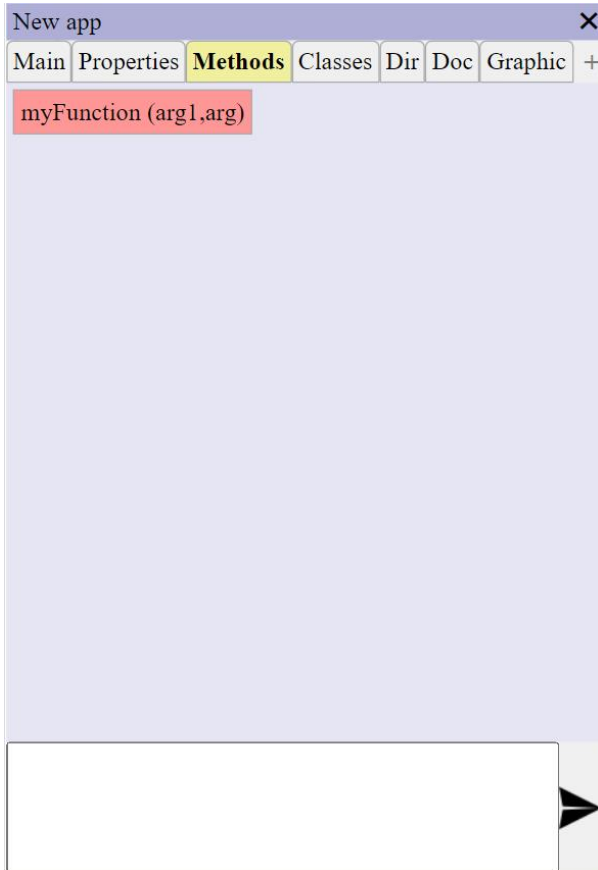


*Création de méthode*

On utilise la commande **fn**

```
fn myFunction(arg1, arg)
```

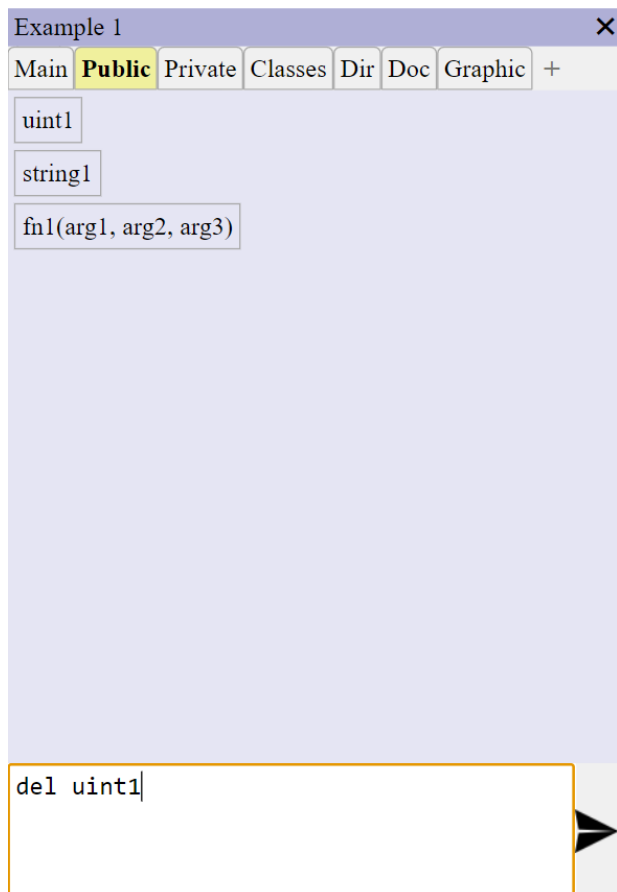
Elle apparaît en rouge, signifiant qu'elle n'est pas encore implémentée :



### *Suppression d'un membre*

On procède par le menu contextuel : clic droit sur l'élément > **Supprimer**

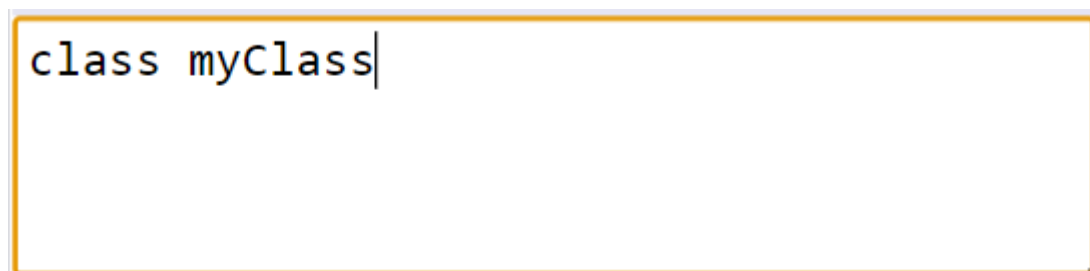
Ou on utilise la commande **del**



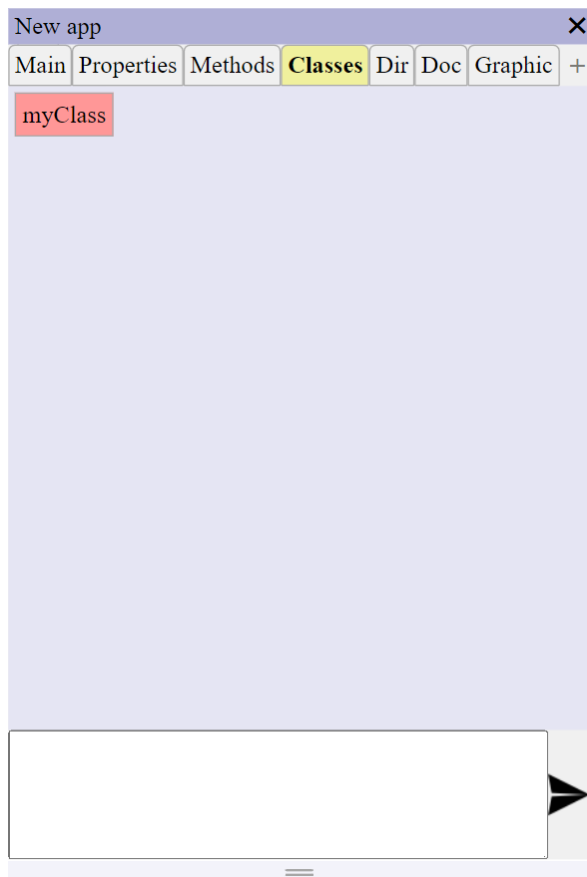
Classes

*Création*

Commande **class**



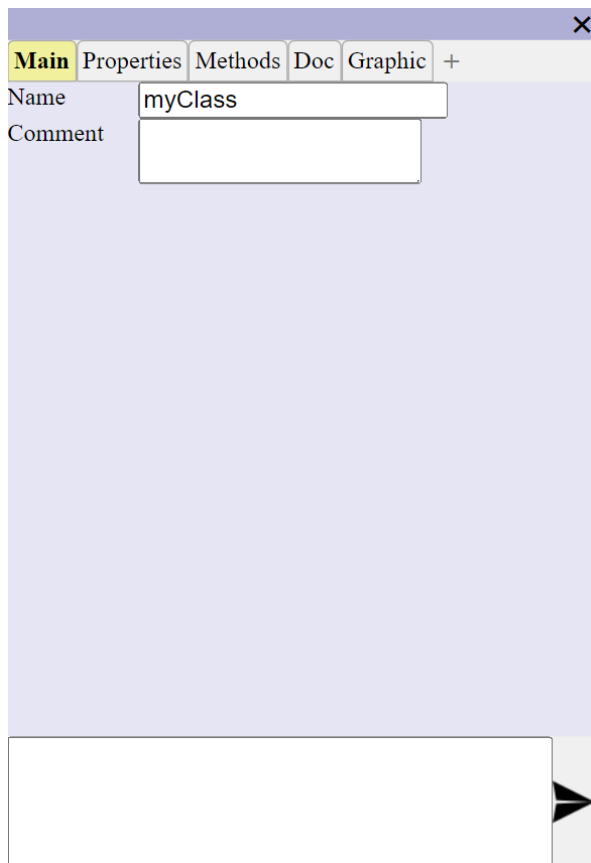
La nouvelle classe apparaît dans l'onglet **Classes** de l'application :



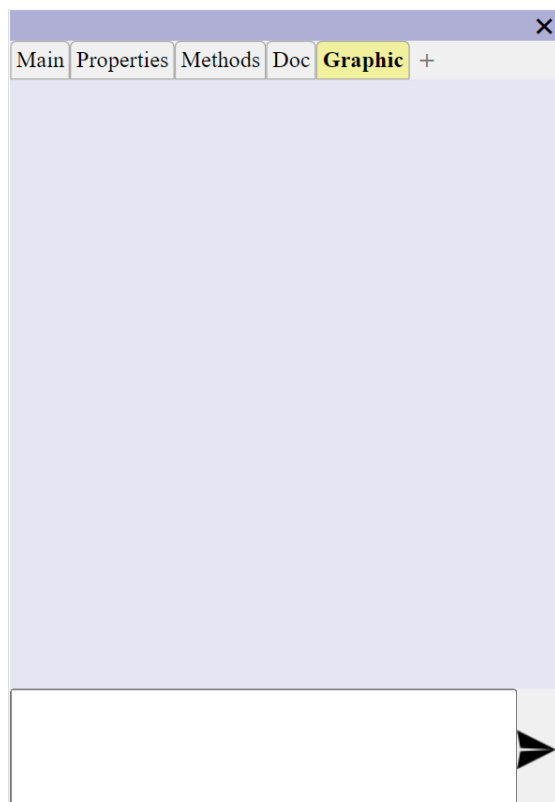
### **Remarque**

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 :



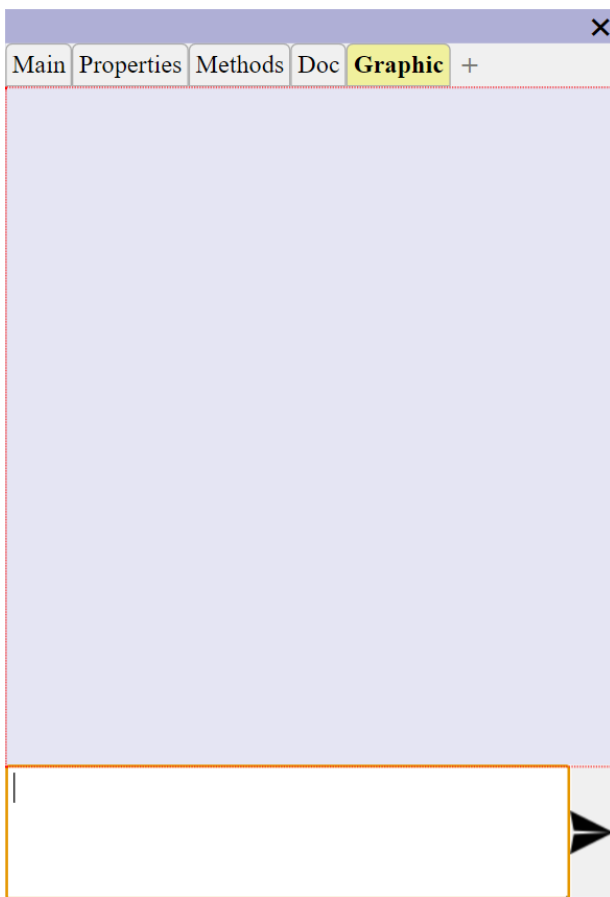
En effet une classe n'est pas nécessairement graphique.

### *Rendre une classe graphique*

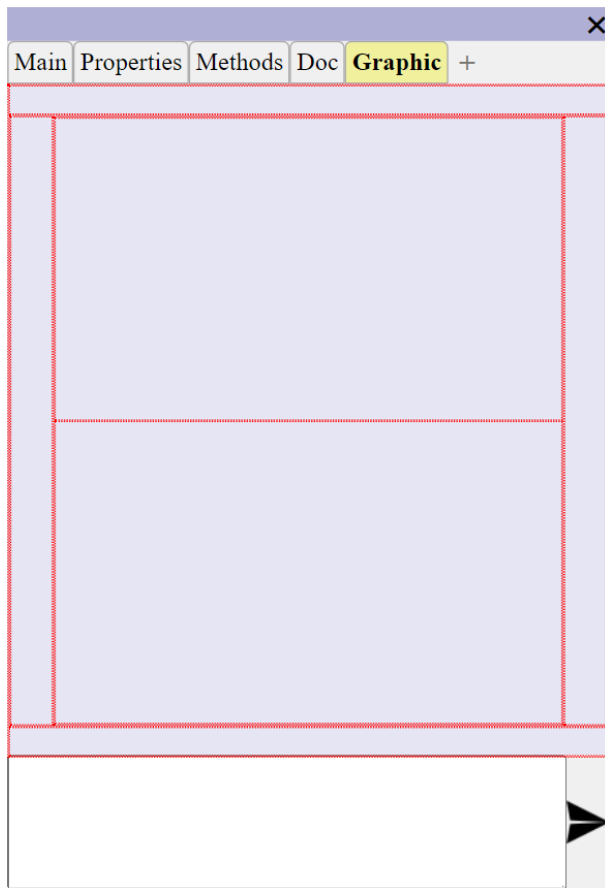
- Taper la commande **graphic** :



Un élément div apparaît encadré de rouge :

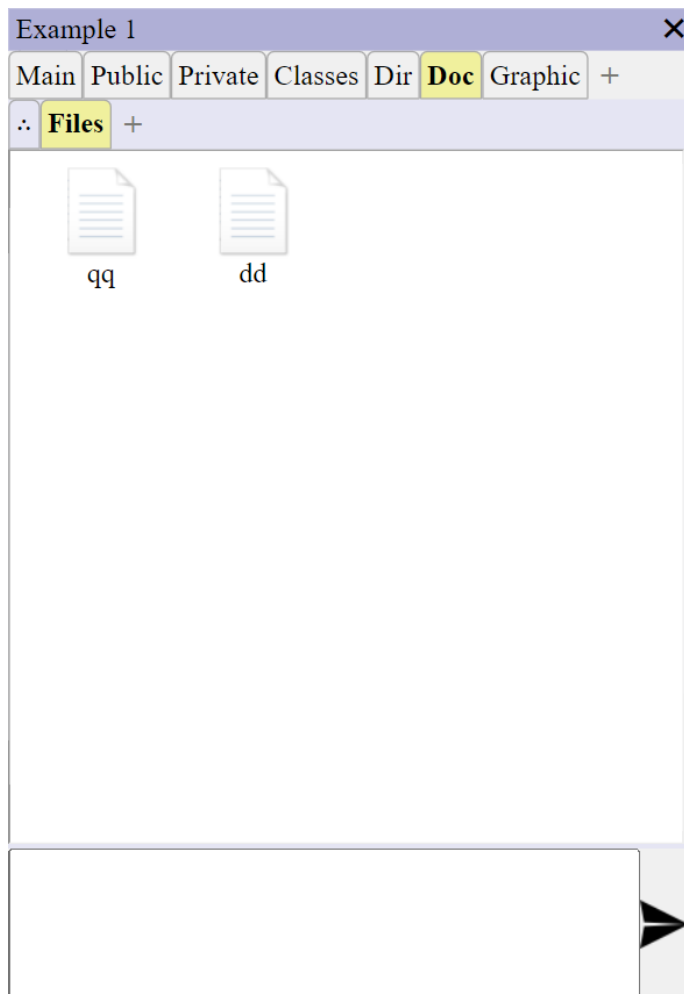


On peut alors modifier ses propriétés y insérer des insérer des éléments enfants :



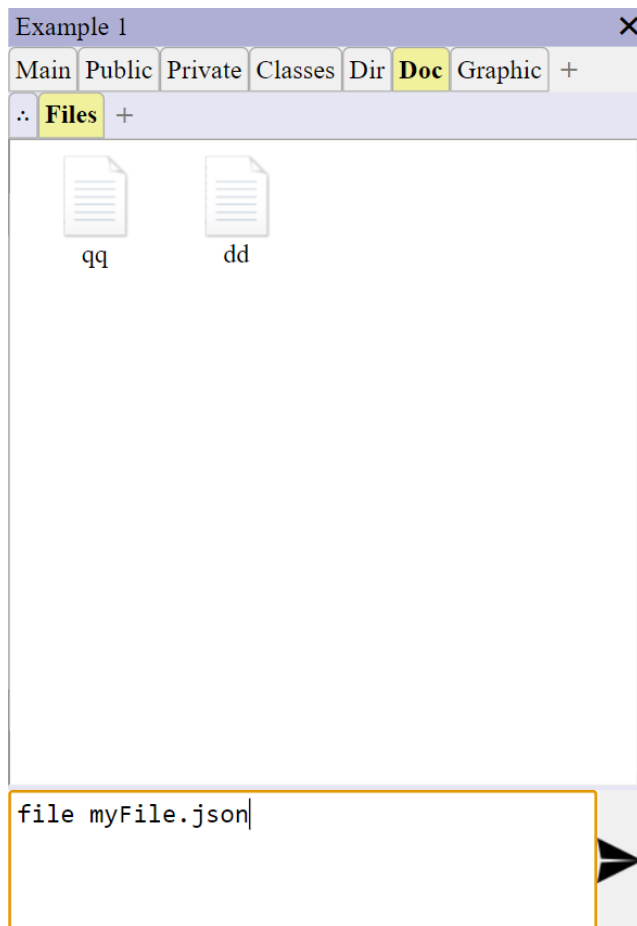
## Gestion de fichiers

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



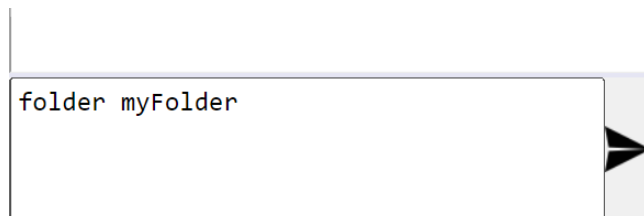
*Création d'un fichier*  
Commande **file**





### *Création d'un dossier*

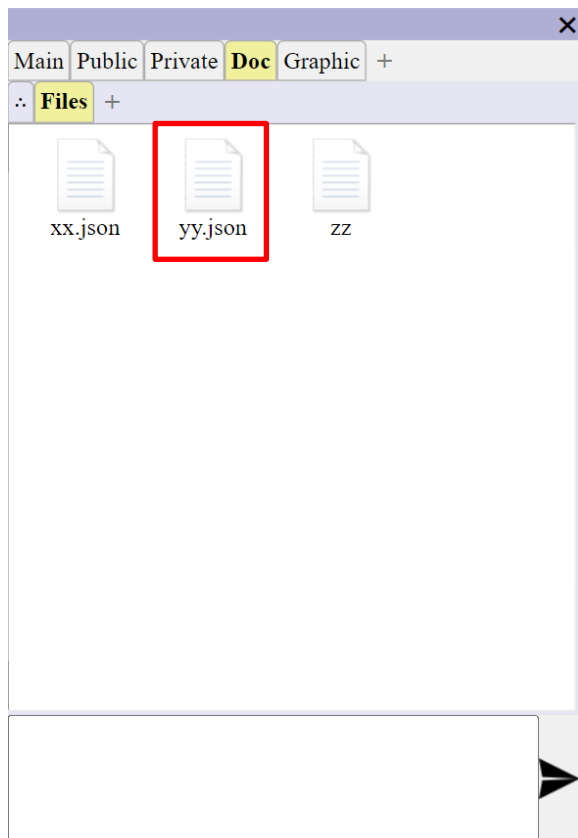
Commande **folder** (ou **fdl**)



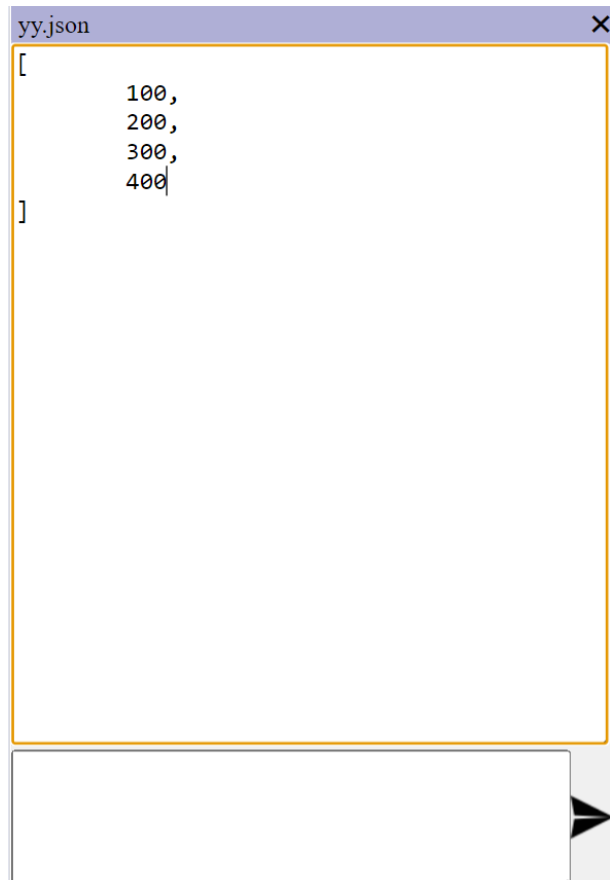
Bien sûr un dossier peut à son tour comprendre d'autres dossiers et des fichiers quelconques

### *Ouverture d'un fichier*

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



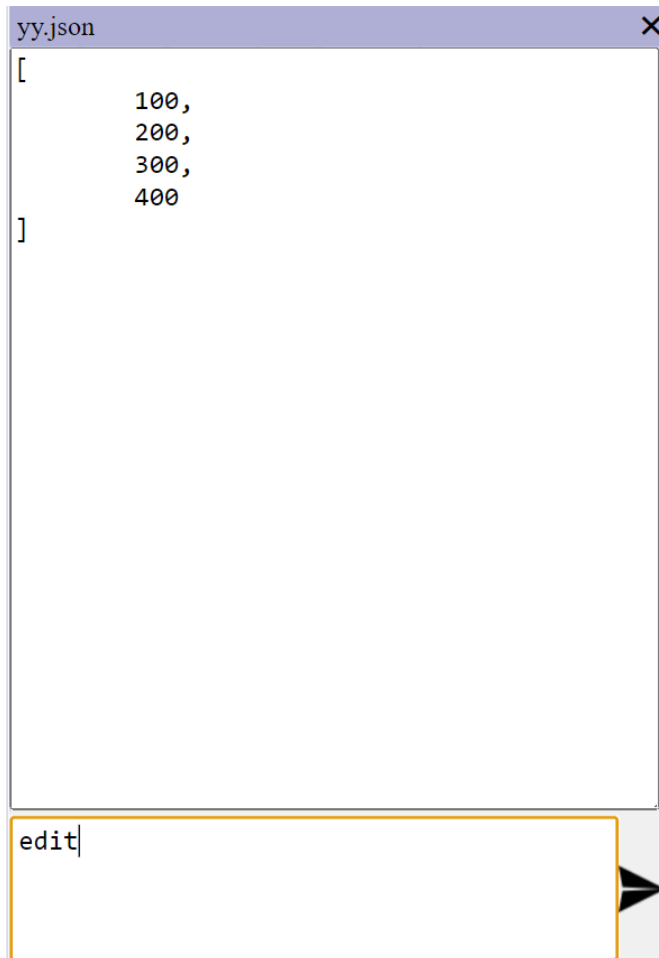
Le nom du fichier apparaît dans la barre de titre :



### *Edition d'un fichier*

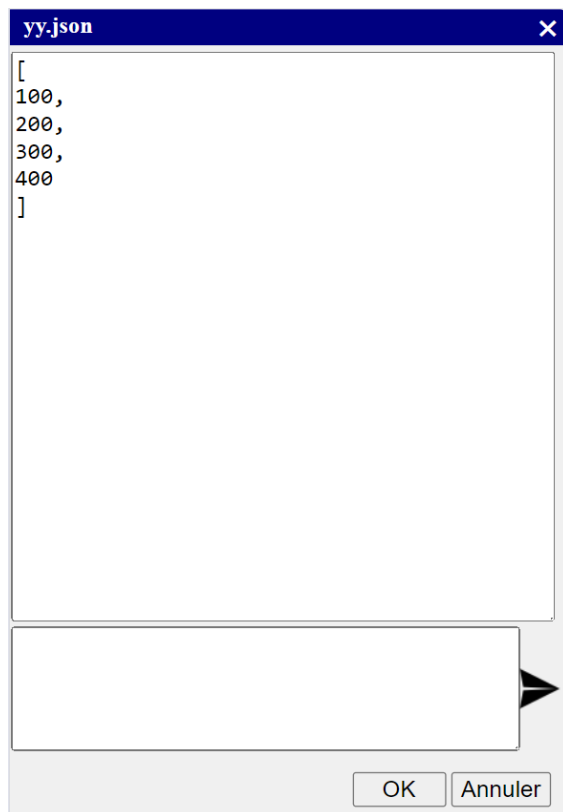
Le fichier étant ouvert,

- Taper la commande ***edit***



```
yy.json ×  
[  
    100,  
    200,  
    300,  
    400  
]  
  
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*)

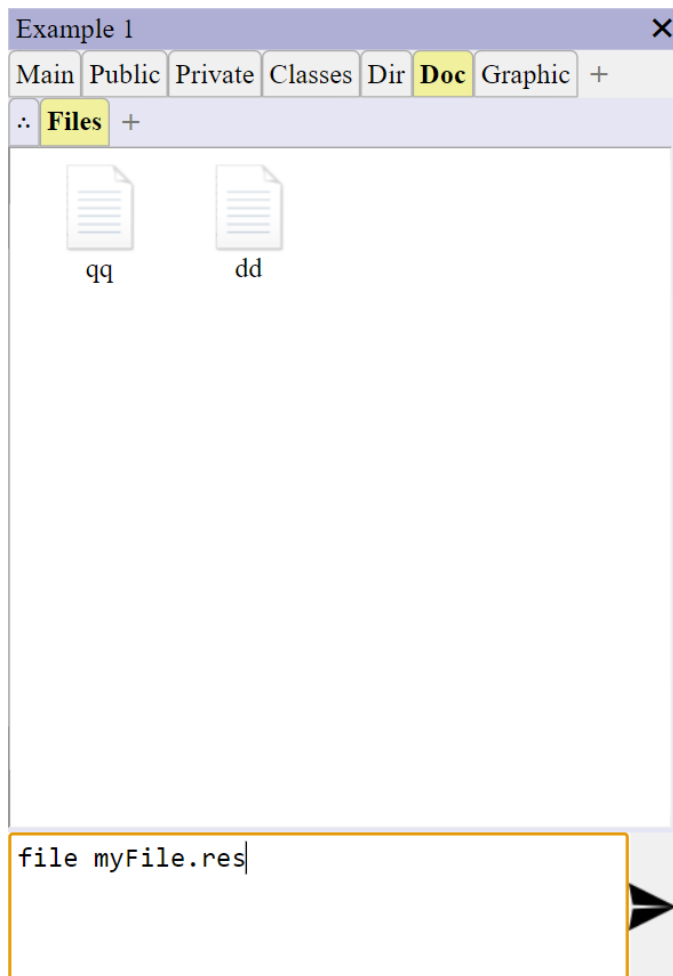


On peut alors éditer son contenu en écrivant directement dans l'éditeur avec les commandes *array*, *obj*, *del*.

### *Créer un fichier json*

Dans un onglet *Doc*

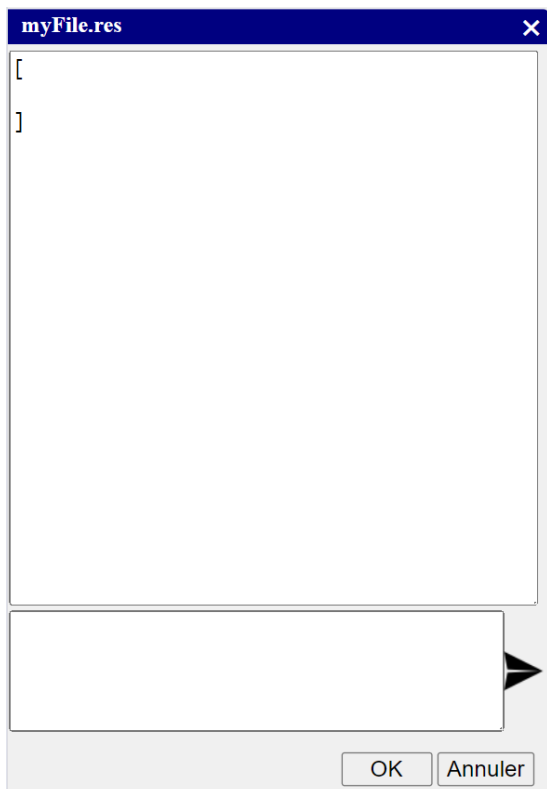
- Créer un fichier quelconque (peu importe l'extension éventuelle) :



- 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** :

```
rename|
```

### Sélection

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

```
selection|
```

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

```
selection multiple
```

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

```
m|
```

On peut désactiver la sélection par la commande **unselect** (ou **unsel** ou **uns**)

```
unselect|
```

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

```
all|
```

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

```
uall|
```

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

```
p|
```

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

```
open|
```

## Copier-coller

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

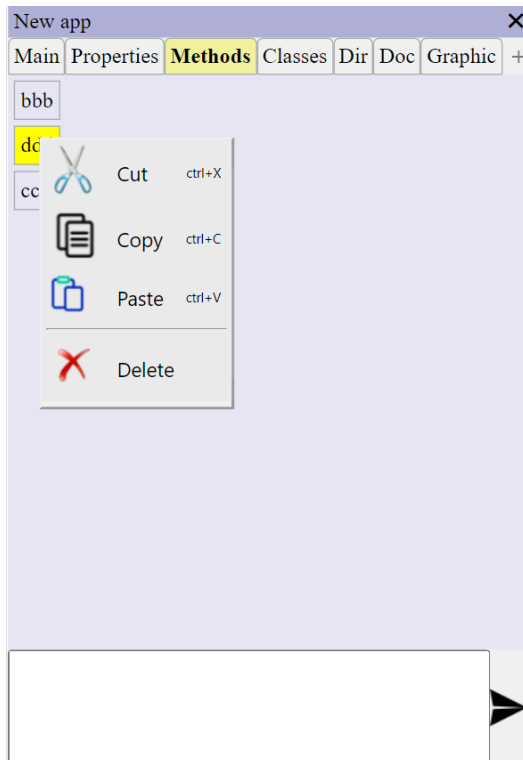
- Les propriétés



- Les méthodes
- Les classes
- Les documents
- Les éléments HTML

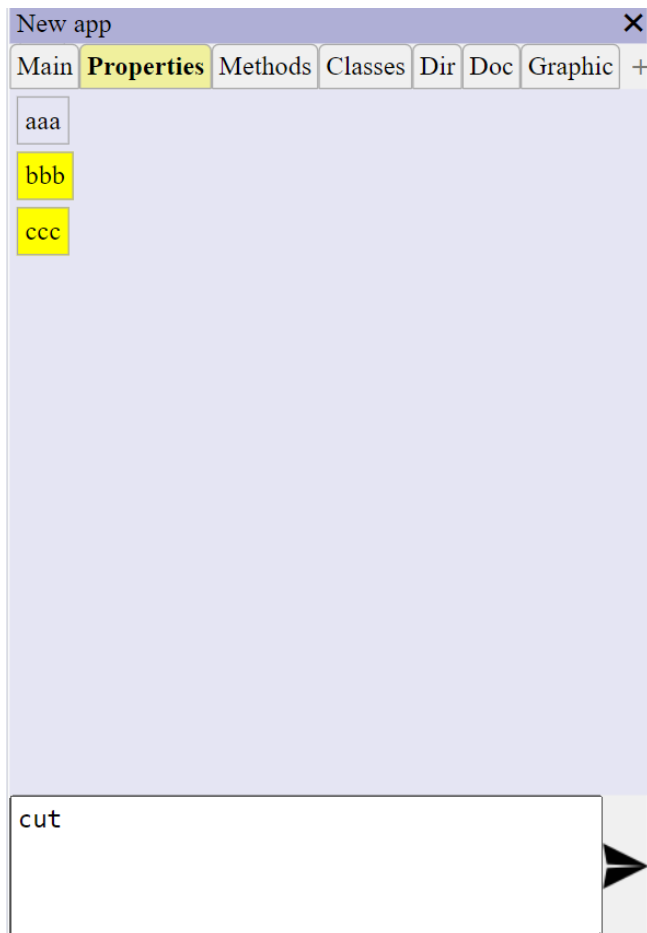
Pour cela,

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



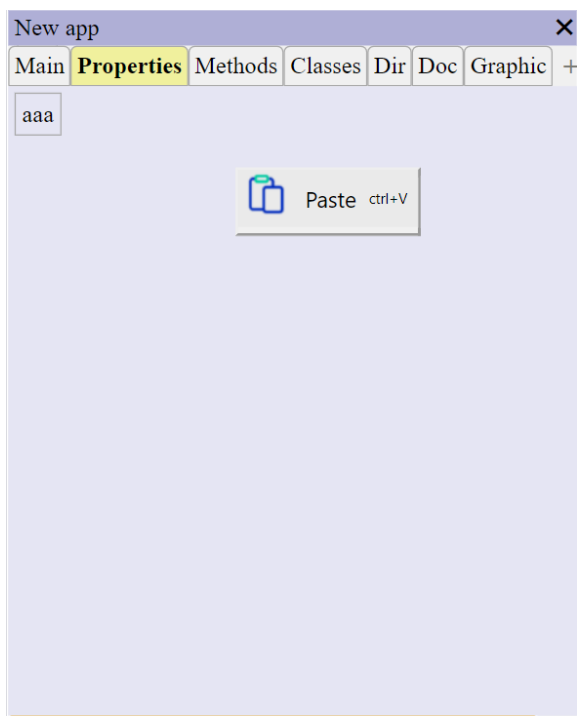
### **Attention**

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** :



## Éléments HTML

### Généraux

- **id** *monId* : affecte l'id *monId*

```
id myId|
```

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

```
cls myClass|
```

- **css**

```
css  
font-size:12pt  
font-style italic  
font weight bold|
```

Les deux points (:) ne sont pas obligatoire (sauf en cas d'ambiguïté) : 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**)

```
style  
font-size : 12pt  
font-style italic  
font weight bold|
```

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

```
attrs
selected:this.getSelected()
enabled:this.enabled|
```

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

- **isElt** *myId*

```
isElt myId|
```

- **isObj** *myId*

```
isObj myId|
```

## 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`

## Marges

- **margin** *10*

- **padding** *10*

## bordures

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

## Couleurs

- **background** (ou **bg**) *red*  
équivalent à *background-color*
- **color** (ou **fg**) *red*  
équivalent à *color*
- **width** *150*  
équivalent à *width*
- **height** *50*  
équivalent à *height*
- **border** *solid 1px red*  
équivalent à *border*

## Contenu texte

- **html** *mon texte*

## Retirer le style par défaut

- **rs** (=Remove Style)

## Autres

- **src**

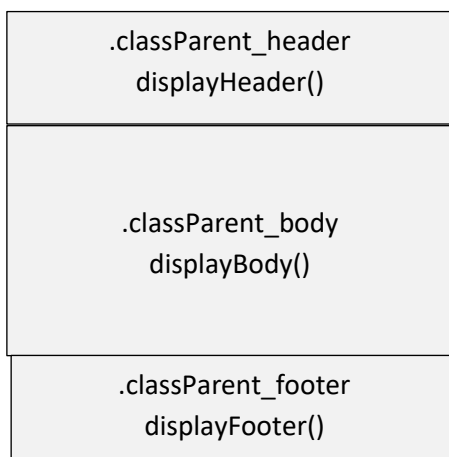
```
src img/monImage.png|
```

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

## Créer des éléments enfants

*Par scission de l'élément courant*

- **v** : scinde verticalement en 3, l'élément central occupant le maximum d'espace (*flex:1*)



- **h** : même chose mais horizontalement

.classParent_left displayLeft()	.classParent_middle displayMiddle()	<i>.classParent_right</i> <i>displayRight()</i>
------------------------------------	--	--

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

.classParent_1	.classParent_2
----------------	----------------

- **h3** : idem en 3 parties égales

.classParent_1	.classParent_2	.classParent_3
----------------	----------------	----------------

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

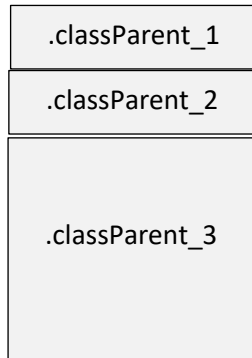
.classParent_1
.classParent_2

- **v3** : idem en 3 parties égales

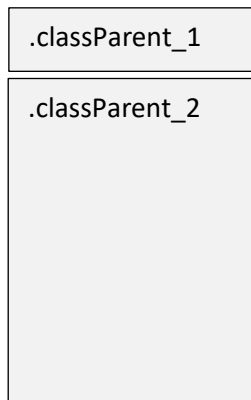
.classParent_1
.classParent_2
.classParent_3&



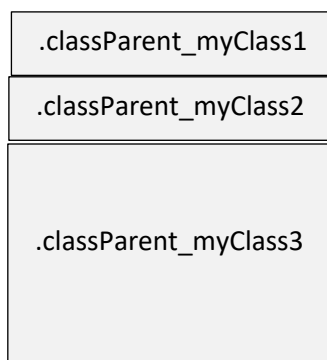
- **v 50px 50px 150px** : idem avec affectation de tailles



- **v 20% 80%**



- **v 50 50 150**  
**myClass1**  
**myClass2**  
**myClass3**



*Par simple ajout*  
**addChild** (ou **child**)

```
child listEmployees
#list1
_list|
employees=appli.getEmployees()
```

Crée un élément avec :

- tag :*listEmployees*
- id :*list1*
- isObj : *\_list*
- employees=*appli.getEmployees()*

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

```
children
div maClasse #id _isElt maFonction
div maClasse2 maFonction2|
```

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

```
children div
aaa #aaa _aaa aaa
|bbb maFonction1
```

Crée 2 éléments 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** :

```
textarea maClasse1 maClasse2
#myTextarea
_textarea
```

Bouton

Commande **button** :

```
button My button
#myButton
_button1
maClasse1 maClasse2
```

img

```
img img/img1.png
maClasse1
#monImage
_image1
```

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 :

```
img download|
```

tabs

```
tabs  
_tabs1|
```

tab

```
tab tab1 Tab 1|
```

### Attention

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

```
div|
```

Element quelconque

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

```
child textarea  
#myTextarea  
_area1  
html:My text|
```

## Ajouter plusieurs éléments

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

- Si tous les éléments doivent être de même tag :

```
children textarea  
classArea1 #myTextarea1 _area1  
classArea2 #myTextarea2 _area2  
classArea3 #myTextarea3 _area3
```

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 :

```
children  
textarea classArea1 #myTextarea1 _area1  
div classDiv2 #myDiv2 _div2  
textarea classArea3 #myTextarea3 _area3
```

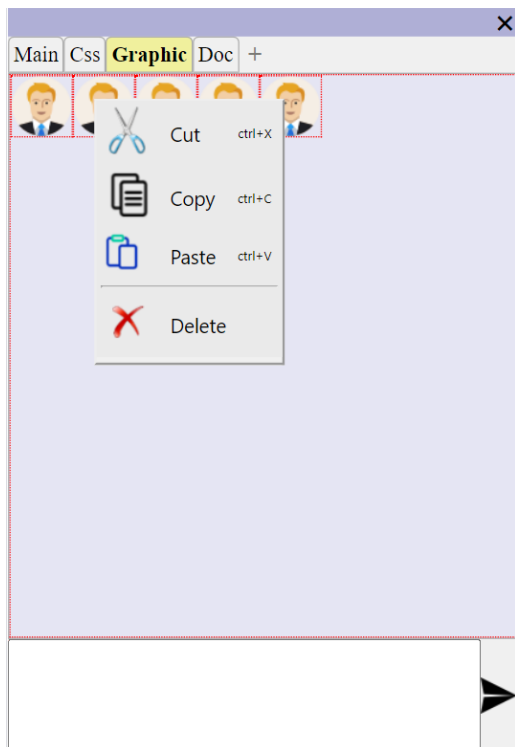
Cette fois chaque ligne commence par le tag suivi de la classe et un nombre variable de paramètres

## Supprimer des éléments

### 1 seul élément

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) élément(s)
- Taper **delete**



### Attention

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

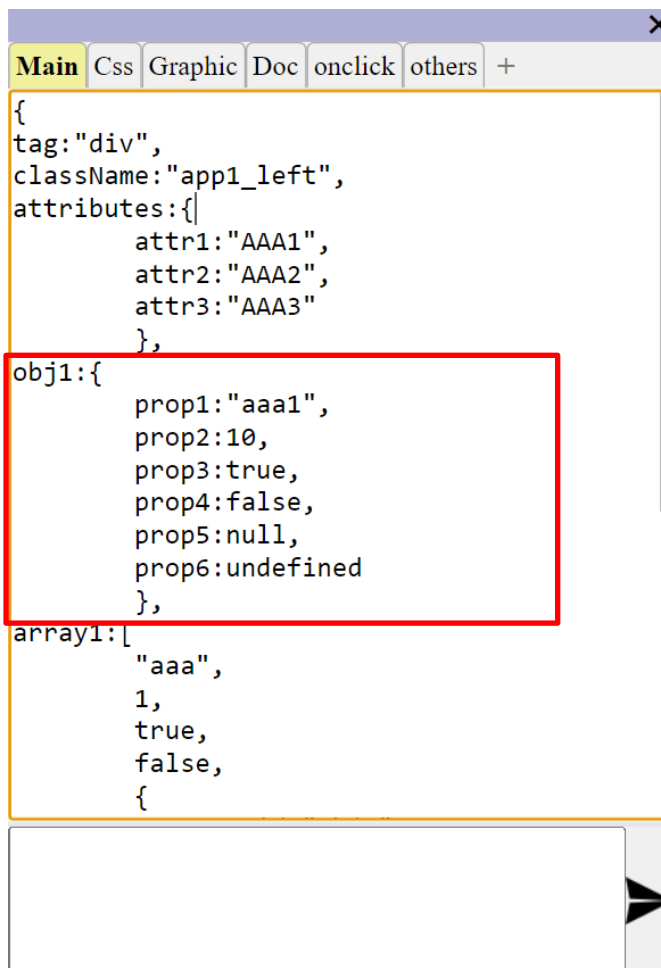
### *Tous les éléments enfants*

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

```
clear|
```

### Objets

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



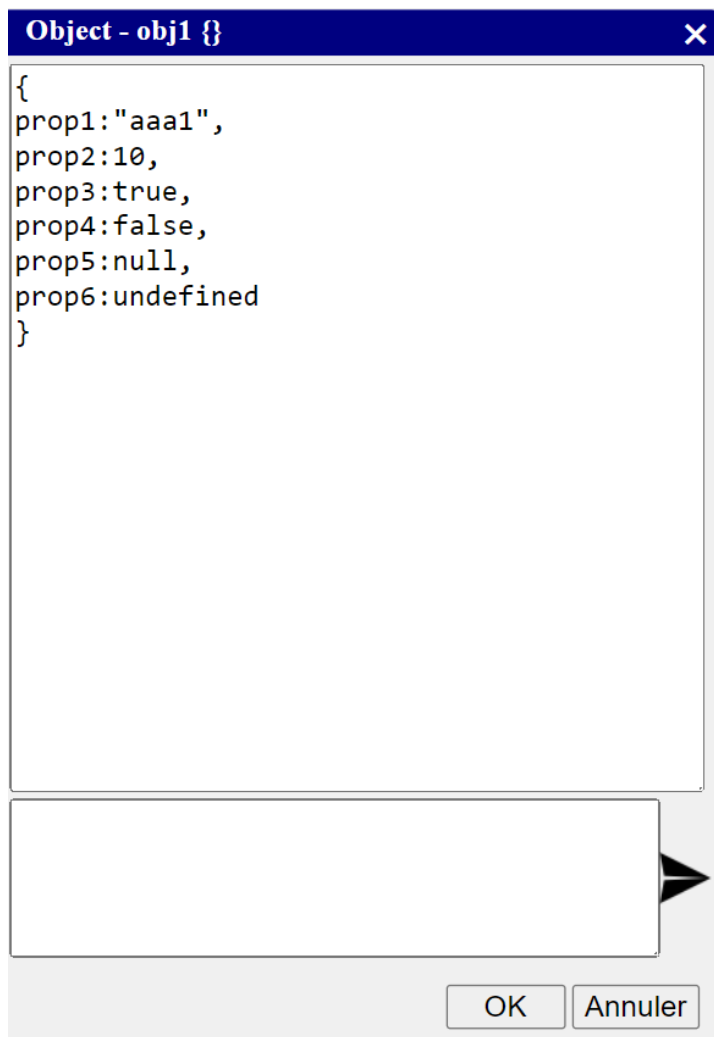
```
{
  tag:"div",
  className:"app1_left",
  attributes:{
    attr1:"AAA1",
    attr2:"AAA2",
    attr3:"AAA3"
  },
  obj1:{
    prop1:"aaa1",
    prop2:10,
    prop3:true,
    prop4:false,
    prop5:null,
    prop6:undefined
  },
  array1:[
    "aaa",
    1,
    true,
    false,
    {
```

### *Edition d'une propriété de type objet*

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

obj obj1|

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 :

obj obj2|

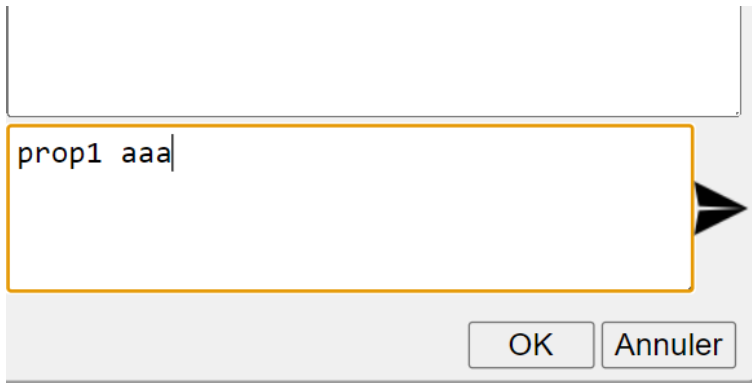
On accède à l'éditeur d'objet vu précédemment

### *Edition de l'objet*

#### Affectation d'une propriété scalaire

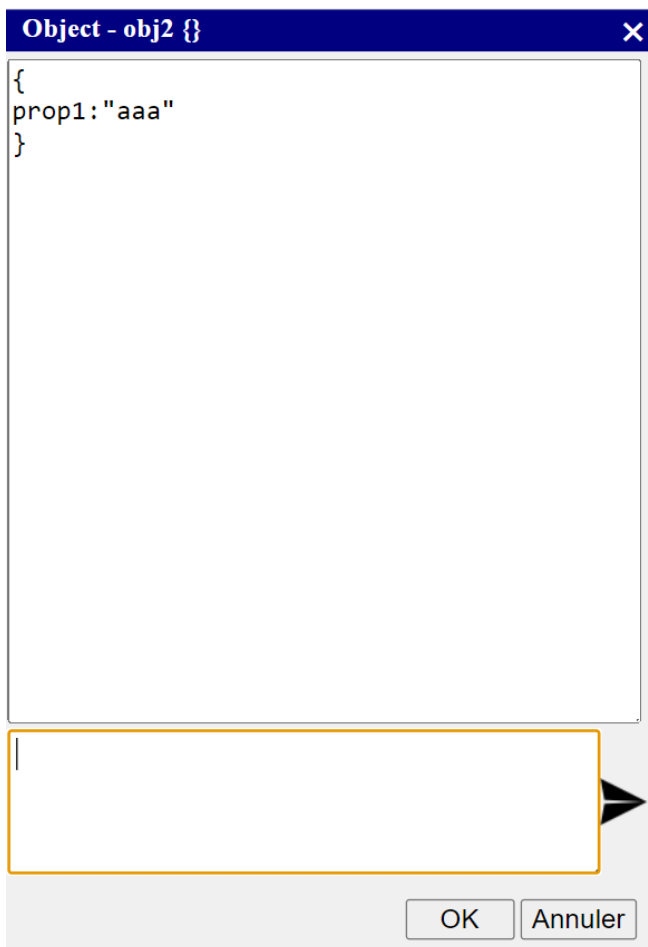
Depuis l'éditeur d'objet, il suffit de taper les propriétés sous la forme :

**nomPropriété valeurPropriété**



A screenshot of a dialog box for editing an object. The dialog has a title bar and a close button. The main area contains a text input field with the text "prop1 aaa" and a cursor at the end. Below the input field are two buttons: "OK" and "Annuler".

L'objet est alors mis à jour :



A screenshot of a dialog box titled "Object - obj2 {}". The main area displays a JSON object: 

```
{  
  prop1: "aaa"  
}
```

. Below the main area is an empty text input field. At the bottom are "OK" and "Annuler" buttons.

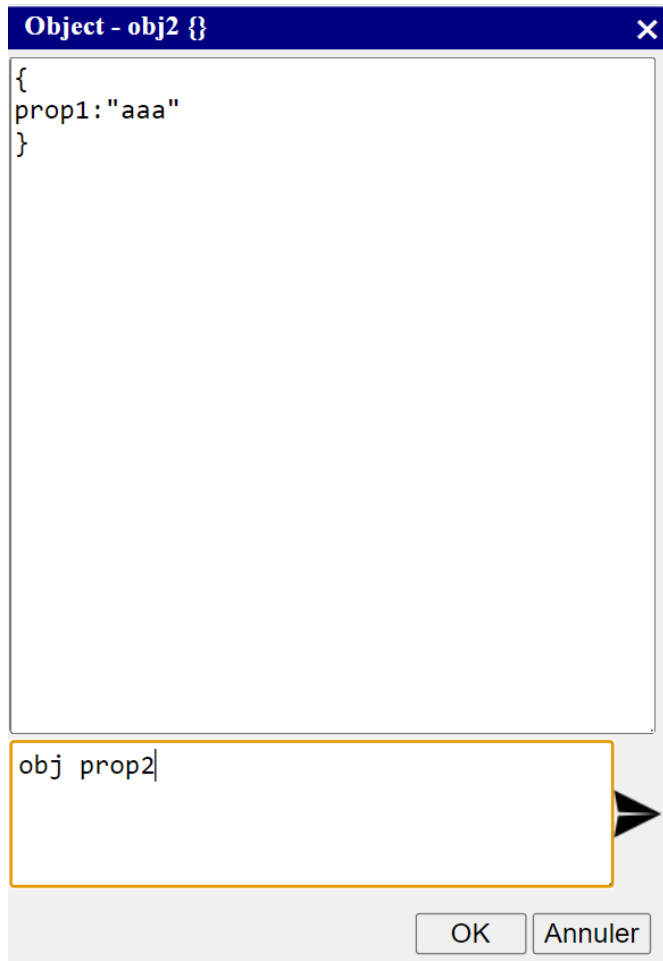
### **Remarque**

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 laquelle on indique le nom de la future propriété :

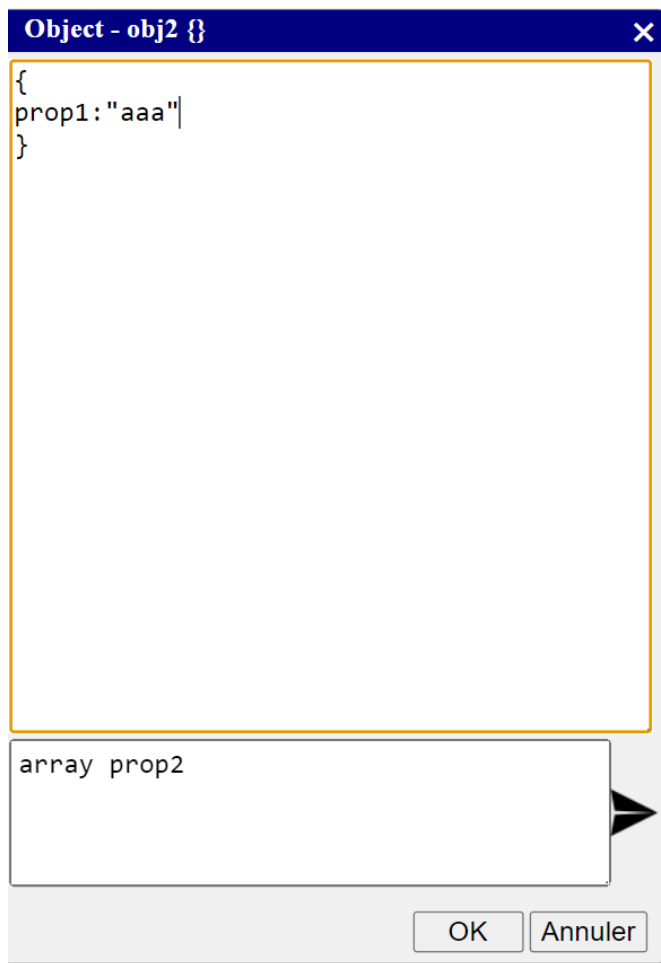


Cela ouvre un nouvel éditeur d'objet auquel on peut ajouter des propriétés.

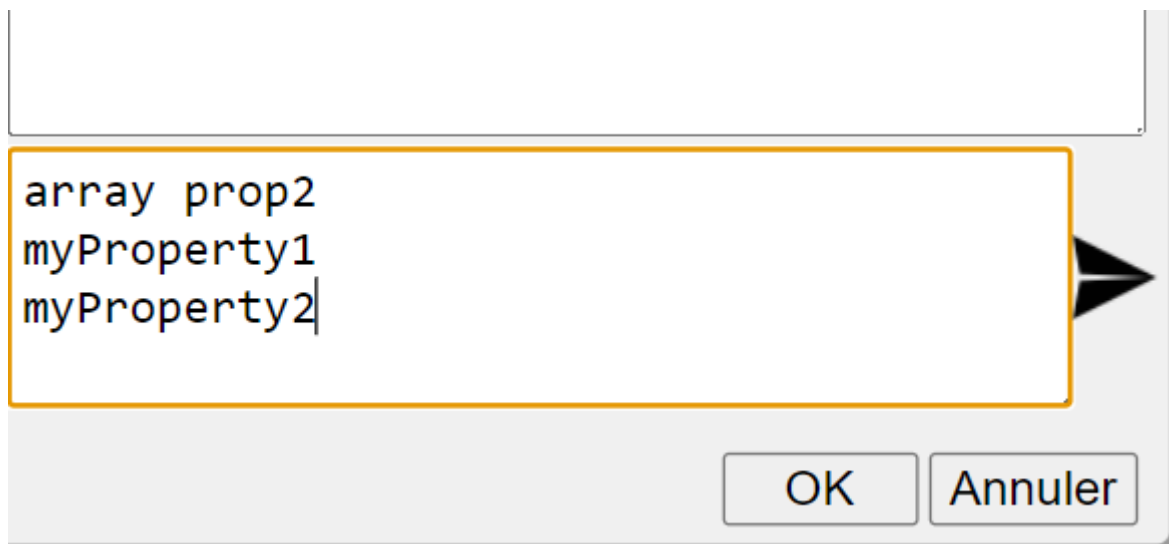
#### Affectation de propriété de type tableau

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

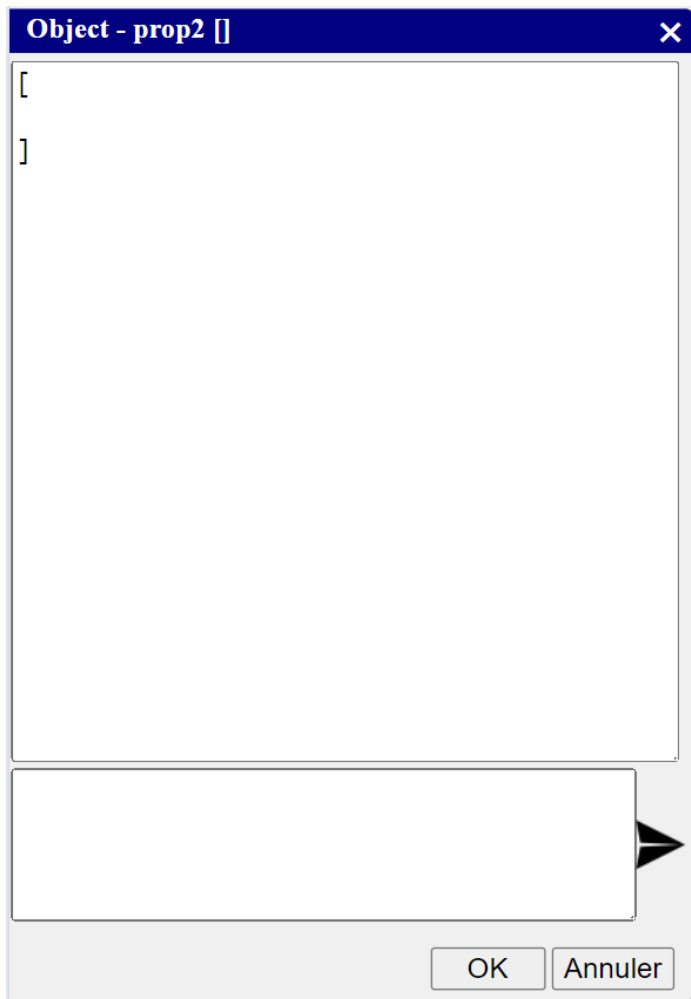




On peut aussi ajouter la liste des propriétés attendues par les futurs enregistrements :



Cela ouvre l'éditeur de tableau :



→ Voir Tableaux.

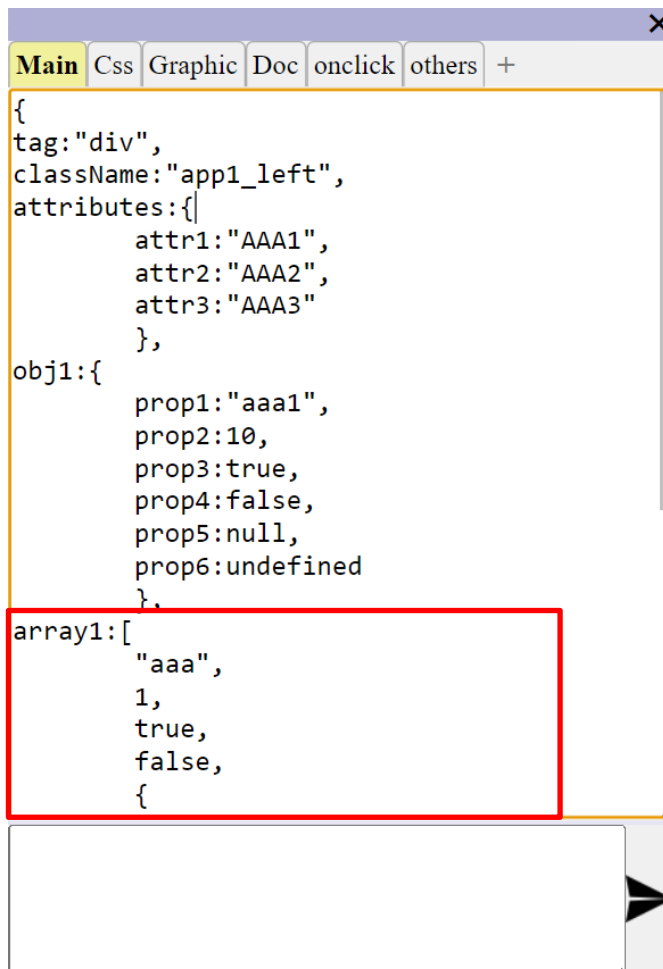
*Supprimer une propriété*

Commande **del**

```
del myProperty|
```

Tableaux

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



The screenshot shows a web editor window with a tab labeled 'Main'. The editor contains a JSON object with the following structure:


```
{
  tag: "div",
  className: "app1_left",
  attributes: {
    attr1: "AAA1",
    attr2: "AAA2",
    attr3: "AAA3"
  },
  obj1: {
    prop1: "aaa1",
    prop2: 10,
    prop3: true,
    prop4: false,
    prop5: null,
    prop6: undefined
  },
  array1: [
    "aaa",
    1,
    true,
    false,
    {

```

The 'array1' property and its contents are highlighted with a red rectangular box. Below the editor is a scroll bar with a right-pointing arrow.

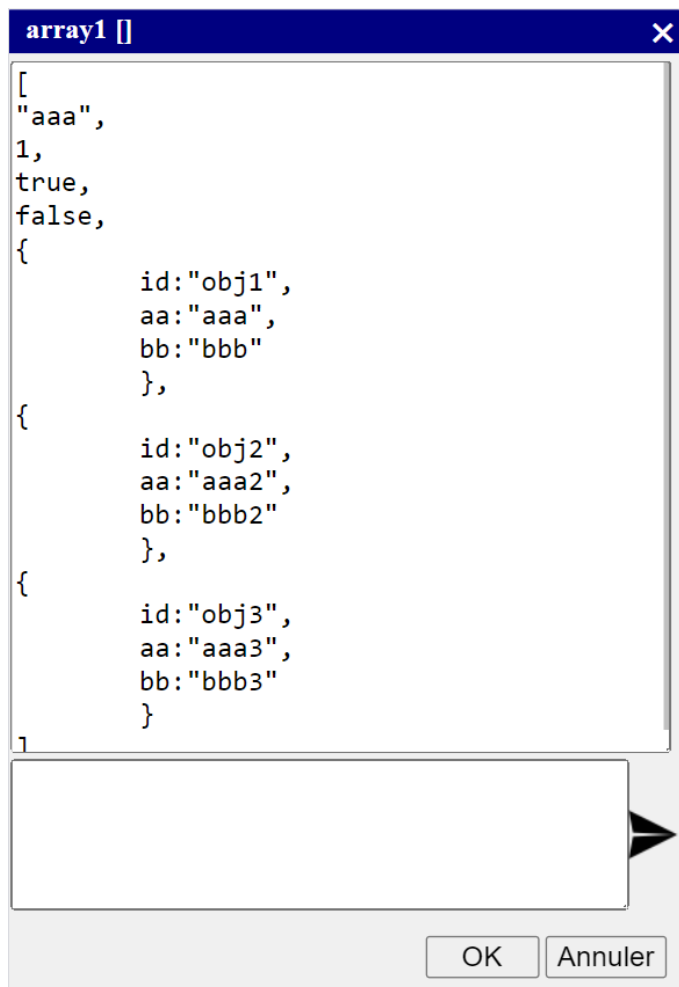
#### *Edition d'une propriété de type tableau*

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



The screenshot shows a text input field with a yellow border. The text 'array array1' is entered into the field, and the cursor is positioned at the end of the text.

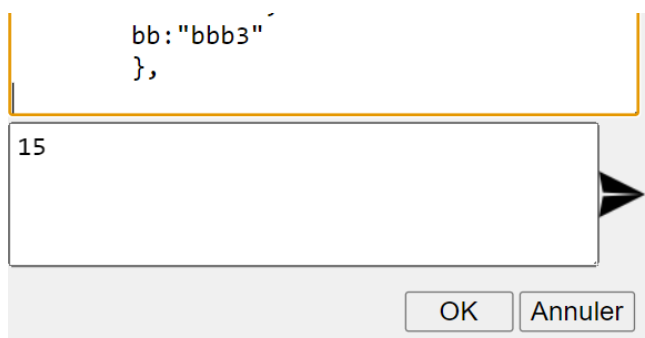
L'éditeur de tableau apparaît :



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

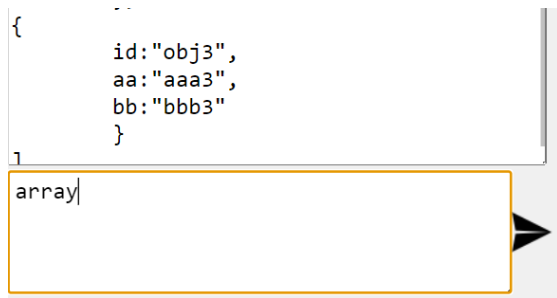
#### Ajout d'un enregistrement scalaire

On peut taper directement la valeur :

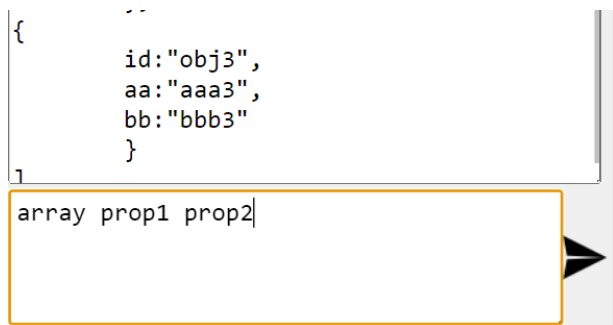


#### Ajout d'un enregistrement de type tableau

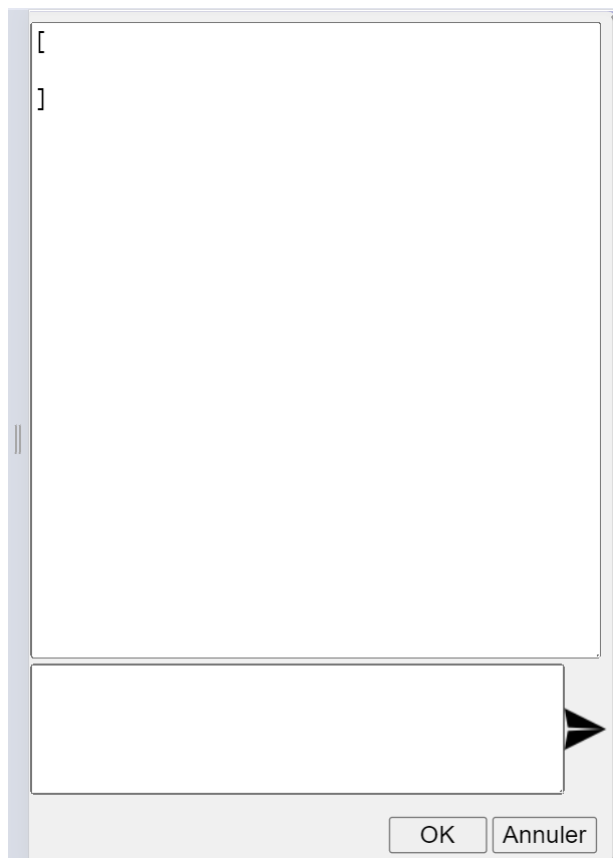
Pour insérer un tableau dans un tableau on utilise la commande **array** :



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
```

```
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 :

```
array array1
property1
property2
property2|
```

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

```
aa: "aaa3",
bb: "bbb3"
},
{
  obj
  aaa
  15
  true
}
```

Supprimer un enregistrement

On utilise la commande **del**

Supprimer le 1<sup>er</sup> enregistrement

```
del first|
```

Supprimer le dernier enregistrement

```
del last|
```

Par numéro d'index

Supprime l'enregistrement numéro **indexEnregistrement** (à partir de 0).

```
del 2|
```

Supprimer tous les enregistrements tels que propriété = valeur

Supprime tous les enregistrements tels que **id=user1**

```
del  
id:user1|
```

Supprimer tous les enregistrements

*del all* (ou *clear*)

```
del all|
```

## Éléments JSA

Form2

*Création*

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

```
form2|
```

```
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* :

```
field address string Address|
```



## 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
```

```
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 défini par la fonction `appli.registerObject()`

```
grid obj1
```

```
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** :

```
field address string Address
```

### Remplissage automatique

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

```
filldata
```

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

```
filldata 100
```

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  
firstName aaa bbb ccc  
lastName AAA BBB CCC  
nb1 10 20 30
```

```
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
cmdFile File
-cmdNew New new.png ctrl+N
-cmdOpen Open open.png ctrl+O
```

```
menubar
#myMenubar
_myMenubar
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 img1.png
cmdUsers Users users.png
cmdGroups Groups groups.png
cmdProperties Properties properties.png
```

```
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**)

```
toolbar
```

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

- **tool** :

```
tool cmdSave save.png Save As...
```

- **combobox** (ou **combo**):

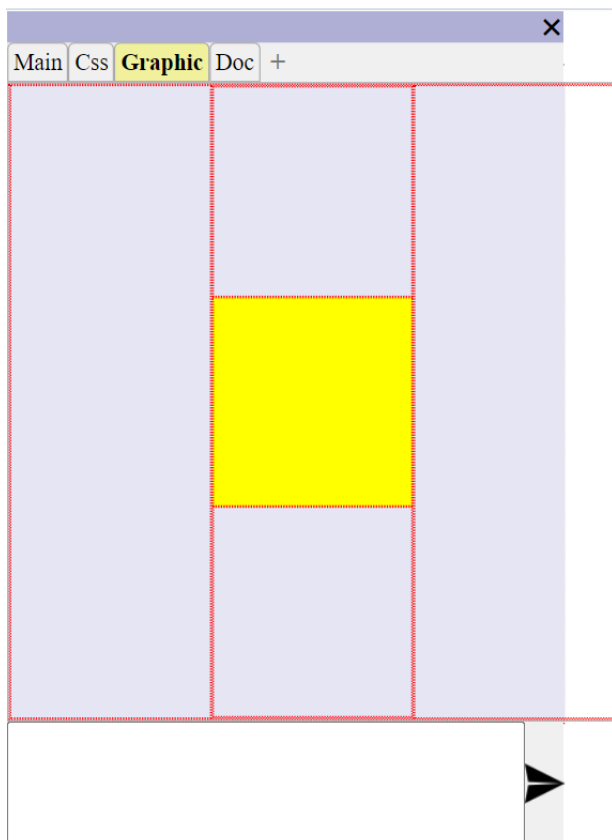
```
combobox cmbList Font-Size
value 1
value 2
value 3
```

Ou par paires clés-valeurs :

```
combobox cmbList1  
key1:Label 1  
key2:Label 2  
key3:Label 3
```

## Déplacement des éléments

Il faut d'abord sélectionner l'élément à déplacer :



Monter

Commande *prev*

```
prev|
```

Descendre

Commande *down*

Monter en premier  
Commande **first**

Descendre en dernier  
Commande **last**