

# Haxetelier 8

Using  
&  
Extern

# Présentation

Jean-Michel Delettre.

Avant : développeur logiciel.

Depuis 2001 : développeur de sites Flash & d'applications web en AS (1 , 2 , 3)

-- Exclusion du FlashPlayer sur mobiles --

Depuis 2 ans : développeur d'applications web mais HTML5 avec le JavaScript correspondant.

... Souhait de continuer avec un langage similaire à AS3  
ou Java :

## Choix de Haxe

Les exemples de cet haxetelier sont en

Haxe  compilé  JS

( avec Api JS récente et standard != js IE<9 )

mais

Haxe étant multi-cibles  
les exemples sont transposables

# Préambule pour $\overline{\text{hx}}$

Haxe a un formalisme typique/classique POO :

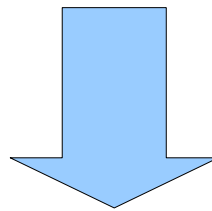
- Une classe est un bloc continu, commençant par le mot-clé « class » (!= fonction)
- L'héritage se fait avec le mot-clé « extends » (!= prototype)
- Les variables et propriétés sont fortement typées ; ainsi que le retour des méthodes.

# [ Typage de variables

- Toutes les variables ont un type (Int, String, MyClass, Date, etc)

déclaré explicitement ou implicitement lors de la 1ère assignation

- A la compilation, Haxe refusera qu'elles reçoivent un contenu d'un autre type ! Le paramètre de retour des fonctions est aussi vérifié.



On va considérer que c'est un gros avantage  
qui corrige plein de coquilles, d'erreurs bêtes  
... et parfois moins bêtes.

]

## « using »

Finalité :

Ajouter des méthodes à une classe existante quand on ne peut pas l'étendre par une nouvelle classe avec « extends »

Exemples :

« Element » de l'api JS

« MovieClip » de Flash si déjà instancié.

Les « String » en dur ...etc.

Néanmoins on veut appeler les méthodes ajoutées avec la même syntaxe que les autres. Soit :

`someInstance.myMethod()`

# « using »

Principe & syntaxe :

Voir « haxetelier8 » \*

Ouvrir : using/src/[StringX.hx](#)

\* sur github : [TODO::github.com/flashline/haxetelier8](https://github.com/flashline/haxetelier8)

## « using » suite

Quelques exemples réels avec « String » :

Si on considère que les strings contiennent de la syntaxe CSS (système utilisé par jQuery ...et JS à présent)

Méthodes ajoutées :

`all()` => renvoie un tableau d'éléments html.

ex: `".someCssClass p".all();`

`get()` => renvoie un seul élément.

ex: `"#myCtnr #myDiv".get().appendChild(elem) ;`

`on()` et `off()` => add ou remove un listener, en lui passant un objet Dynamic ou typé (typDef) comme paramètre.  
N'existe pas en JS ni en As3.

ex: `"div.menu".on("click",someListener,someTypDef) ;`

`// pose un listener sur toutes les div de classe css "menu"`

`slider()` => équivalent à `new Slider`

ex: `"#sliderCtnrId".slider(); // renvoie l'instance de Slider`  
au lieu de

`var s= new Slider();`


`s.into="#sliderCtnrId" ;`

Voir sources : `using/src/apix/common/util/StringExtender.hx`  
`using/src/Main.hx`  
`UICompo/src/Main.hx`



Exemple d'abstraction avec « using »  
quand le code est dépendant de la cible (js,flash,etc) :

```
#if js
    import js.html.Element;
    using apix.common.display.ElementExtender;
    typedef Elem = Element;
#else if flash
    import flash.display.Sprite;
    using apix.common.display.SpriteExtender;
    typedef Elem = Sprite;
#else
// TODO
#endif
```



```
// utiliser Elem dans le code
// avec les méthodes des xxxExtender ou méthodes communes.
```

Voir sources :    [using/src/apix/common/display/ElementExtender.hx](#)  
                  [using/src/apix/ui/display/Slider.hx](#)

Note: Ceci est une piste de réflexion...OpenFL ou Cocktail gère ce problème avec une api commune. ex OpenFL rend l'api flash compilable pour toutes les plateformes ; et Cocktail :  
html/js => flash => OpenFI

# « extern »

## Finalité :

Utiliser une API native du langage cible,

ex en JS : **jQuery**

(pour laquelle plusieurs « extern » ont déjà été créées),

... sans avoir à l'implémenter en haxe,

tout en bénéficiant de ses avantages dont le fameux

**typage fort !**

note: Plein d' « extern » sont sur : <http://lib.haxe.org/all>

« extern »

Principe ..... avec une API 3D en JS :

Babylon\*

Voir :

[extern/bin/meshTransform.html](http://extern/bin/meshTransform.html)

\* de David Catuhe @deltakosh <https://github.com/BabylonJS/Babylon.js>

download de l'extern : <https://github.com/flashline/Babylon-X>

FIN

Haxe +

...