

# Haxetelier 8

Using  
&  
Extern

# Présentation

Jean-Michel Delettre.  
développeur

Depuis 2002 :  
spécialisé d'applications web,  
en Flash et AS (1 , 2 , 3)

-- Exclusion du FlashPlayer sur mobiles --

Depuis 2 ans :  
développeur d'applications web mais HTML5  
et JavaScript moderne.

... Souhaitant continuer avec un langage similaire à AS3 ou Java :

## Choix de Haxe

Les exemples de cet haxetelier sont en

Haxe  compilé  JS

( Haxe3 + Api JS récente et standard != js IE<9 )

Haxe étant multi-cibles  
les exemples sont transposables

# Préambule pour $\bar{h}x$

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.

[

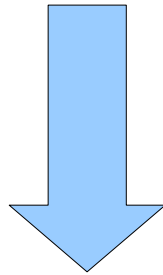
# Typage de variables

- 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 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 », car on ne contrôle pas la création des instances.

Exemples :

« Element » de l'api JS existant dans le .html

« MovieClip » de Flash existant dans le .fla.

Les « String » , les « Float », en dur ...etc.

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

`someNativeInstance.myMethod()`

## « using » suite

Nous allons voir un Exemple réel avec « String » ,

où je considère que les chaines de caractères contiennent de la syntaxe CSS

ex:

`".someCssClass p"` :

Tous les `p` inclus dans toutes les balises de classe css `someCssClass`

`"#myCtnr #myDivId"` :

L'élément d'id `myDivId` se trouvant dans celui d'id `#myCtnr`

`"div.menu , ul.menu"` :

Les éléments de classe `menu` étant des `div` ou `ul` (liste).  
mais pas de `p` ou `span` par exemple.

note:

système très précis de ciblage, déjà utilisé par jQuery  
...et JS à présent.

## « using » suite

Méthodes ajoutées :

`get()` => renvoie un seul élément HTML.

ex:

```
"#myDivId".get().appendChild(elem) ;
```

qui me renvoie l'Element d'id myDivId.

```
"#myCtnrId .h1".get().textContent=someTitle ;
```

qui va assigner le 1er <h1> se trouvant dans le conteneur de l'appli.



## « using » suite

Méthodes ajoutées (suite):

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

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

## « using » suite

Méthodes ajoutées (suite) :

**on()** => équivalent de `addEventListener`  
(soit ajout d'une fonction qui sera exécutée  
sur un événement.)

... en plus court  
et en pouvant passer au listener un paramètre

(ce paramètre est un objet Dynamic ou typé,  
simples paires `label:data` ou structure plus complexe voire  
une instance d'une classe Model)

Note: N'existe pas en JS ni en AS mais en jQuery !

ex:

```
"div.menu".on("click",someListener,false,someObject) ;
```

qui pose un listener sur toutes les div de classe css "menu"  
et lui passe un objet (ou mieux Typdef pour les haxeurs).

**off()** => c'est l'inverse, à savoir  
`removeEventListener()`

sauf que aucun paramètre n'est passé.

# « using » suite

Méthodes ajoutées (suite) :

note: Idée non généralisée  
que je ne suis pas sûr de continuer.

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

au lieu de

```
var s= new Slider();  
s.into="#sliderCtnrId" ;
```

on fait:

```
[var s=] "#sliderCtnrId".slider();
```

```
// renvoie aussi l'instance du Slider  
// le slider est visible dans la page.
```

nb: api vidéo de mozilla

Avant syntaxe et principe avec les sources

Break github et haxe

[facultatif]

odp/pdf et programmes ,  
tout est sur gitHub:

<https://github.com/flashline/haxetelier8> >  
[download zip]

pas de dépendance...

...sauf installer Haxe :

<http://haxe.org/download/>

# « using » suite

Principe et syntaxe :

Ouvrir prog appelant :

**samples/using/src/Main.hx**

puis la classe :

samples/classes/Apix/apix/common/util/**StringExtender.hx**

puis (facultatif) :

samples/UICompo/src/**Main.hx**

Exemple d'abstraction avec « using »  
un même code pour plusieurs 'api' cibles (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;
    //to continue !
#else
    // TODO
#endif
```

// utiliser **Elem** dans le code

// avec les méthodes des **xxxExtender** (ou méthodes communes si existent)

cas : addChild() vs appendChild() ).

Voir sources :

sample/classes/Apix/apix/ui/slider/**Slider.hx**  
dont elem.addChild()

[sample/classes/Apix/apix/common/display/**ElementExtender.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 => OpenFL

# « extern »

## Finalité :

Utiliser une API native du langage cible,

ex en JS : **jQuery**

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

... sans avoir à la ré-écrire 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\*

\* de David Catuhe

@deltakosh

<http://www.babylonjs.com/>

download de l'extern :

<https://github.com/flashline/Babylon-X>

(existe déjà dans github haxetelier8)



# "extern" suite

Voir :

<samples/extern/bin/meshTransform.html>

puis l'extern :

<samples/classes/Babylon/babylonx/cameras/Camera.hx>

puis le programme appelant :

<samples/extern/src/cameras/MeshTransform.hx>

ou [Test.hx](#)

# "extern" suite

Les exécutable :

[http://localhost/\\_\\_\\_HAXE/haxetelier%208/samples/extern/bin/](http://localhost/___HAXE/haxetelier%208/samples/extern/bin/)

ou en ligne :

<http://www.pixaline.net/intra/3djs/>

# FIN

...prochain sujet :  
Apache [Cordova](#)  
(~ libre de Adobe [PhoneGap](#))

création d'applications mobiles hybrides.

soit le cycle complet :  
[Haxe->JS->Cordova->Android playStore](#)

(API très fiable !)

[ questions ? ]

# Haxe +