# Duik

### //TODO explications low level, mid level, high level des méthodes

## Attributes

string *Duik.version*
float *Duik.versionNumber*
boolean *Duik.forceReload*

| Name | Type | Description |
|---|---|---|
| ***version*** | string, read-only | Version string of libDuik |
| ***versionNumber*** | float, read-only | Version number of libDuik |
| ***forceReload*** | boolean, read-only during run time | When true, forces libDuik to be reloaded each time it is included in a script.<br>When false, libDuik loads only on first run and then stays available until After Effects is shut down.<br><br>This attribute should not be changed, unless you're editing libDuik.jsxinc itself and you need to do some testing and reload it without shutting down After Effects.<br><br>To speed up launching of your scripts even at first launch, instead of including *libDuik.jsxinc*, you can copy it in the *Startup* subfolder of the *Scripts* folder of After Effects, and rename it to *libDuik.jsx*. This way, libDuik will be available to all scripts without *#include* macro.<br><br>This attribute must be set by directly by editing libDuik.jsxinc to work. |

## Classes

*Duik.uiString*
*Duik.settings*
*Duik.utils*

| Name | Description |
|---|---|
| ***uiStrings*** | Contains all string names used by effects created by Duik.<br>You can set these strings to translate libDuik at runtime.<br>Default values are English names. |
| ***settings*** | Access to settings used by Duik. |
| ***utils*** | Some useful tools |

## Methods

**//TODO tri par level**

**Low-level methods are listed below (greyed) but they are not documented.**
**If you do not understand what low-level methods do by reading them in *libDuik.jsxinc*, you shouldn't need them.**

*Duik.addIK(controller, layer1, layer2, layer3, goal, clockWise, threeD, frontFacing)*
*Duik.addGoal(layer, controller)*
*Duik.addController(layer)*
*Duik.addControllers(layers)*
*Duik.addOneLayerIK(controller,layer)*
*Duik.addTwoLayerIK(threeD,controller,root,end,clockWise,frontFacing)*
*Duik.addWiggle(layer,property,all,x,y,z)*
*Duik.add3DWiggle(layer,property,x,y,z)*
*Duik.add2DWiggle(layer,property,x,y)*
*Duik.add1DWiggle(layer,property)*

| Name | Description | Return | Level |
|---|---|---|---|
| ***addIK(controller, layer1, layer2, layer3, goal, clockWise, threeD, frontFacing)*** | Adds IK on the layers | true if successful, false if anything went wrong | mid |
| ***addGoal(layer, controller)*** | Adds a goal effect to the layer, which may be controlled by a controller | true if successful, false if anything went wrong | mid |
| ***addController(layer)*** | Creates a null object (controller) at layer position and named by layer.name | AVLayer; controller | mid |
| ***addControllers(layers)*** | For each layer,<br>Creates a null object (controller) at layer position and named by layer.name | Array of AVLayer; controllers | High |
| ***addWiggle(layer, property, all, x, y, z)*** | Adds a wiggle effect to given property | true if successful, false if anything went wrong | mid |

***Duik.addIK(controller, layer1, layer2, layer3, goal, clockWise, threeD, frontFacing)***

*mid-level method.*

Adds IK on the layers

parameters:

controller | AVLayer
layer1 | AVLayer
layer2 | AVLayer or undefined
layer3 | AVLayer or undefined

goal | AVLayer or undefined
clockWise | boolean, used only with two-layer and three-layer IK, default: false
threeD | boolean, works only with two-layer IK, default: false
frontFacing | boolean, default: false

returns

true if successful, false if anything went wrong

### ***Duik.addGoal(layer, controller)***

*mid-level method.*

Adds a goal effect to the layer, which may be controlled by a controller

parameters:

layer    | AVLayer
controller | AVLayer or undefined

returns

true if successful, false if anything went wrong

### ***Duik.addController(layer)***

*mid-level method.*

Creates a null object (controller) at layer position and named by layer.name

parameters

layer | AVLayer

returns

AVLayer controller

### ***Duik.addControllers(layers)***

*high-level method.*

For each layer,
Creates a null object (controller) at layer position and named by layer.name

parameters

layers | Array of AVLayer

returns

Array of AVLayer controllers

//TODO A COMPLETER

# Duik.uiStrings

Contains all string names used by effects created by Duik.
You can set these strings to translate libDuik at runtime.
Default values are English names.

## Attributes

*Duik.uiStrings.ikFkBox*
*Duik.uiStrings.reverseBox*
*Duik.uiStrings.fkAngle*
*Duik.uiStrings.rootFkAngle*
*Duik.uiStrings.endFkAngle*
*Duik.uiStrings.goalFkAngle*
*Duik.uiStrings.weightSlider*
*Duik.uiStrings.xAmp = "X Amp.";*
*Duik.uiStrings.xFreq = "X Freq.";*
*Duik.uiStrings.yAmp = "Y Amp.";*
*Duik.uiStrings.yFreq = "Y Freq.";*
*Duik.uiStrings.zAmp = "Z Amp.";*
*Duik.uiStrings.zFreq = "Z Freq.";*
*Duik.uiStrings.amp = "Amp.";*
*Duik.uiStrings.freq = "Freq.";*

| Name | Type | Description |
|---|---|---|
| ***ikFkBox*** | string | *"IK / FK"* |
| ***reverseBox*** | string | *"Reverse"* |
| ***fkAngle*** | string | *"FK"* |
| ***rootFkAngle*** | string | *"Root FK"* |
| ***endFkAngle*** | string | *"End FK"* |
| ***goalFkAngle*** | string | *"Goal FK"* |
| ***weightSlider*** | string | "*LookAt Weight"* |
| ***xAmp*** | string | *"X Amp."* |
| ***xFreq*** | string | *"X Freq."* |
| ***yAmp*** | string | *"Y Amp."* |
| ***yFreq*** | string | *"Y Freq."* |
| ***zAmp*** | string | *"Z Amp."* |
| ***zFreq*** | string | *"Z Freq."* |
| ***amp*** | string | *"Amp."* |
| ***freq*** | string | *"Freq."* |

## Duik.settings

Access to settings used by Duik.

### Attributes

*Duik.settings.controllerSize*
*Duik.settings.controllerSizeAuto*
*Duik.settings.controllerSizeHint*

| Name | Type | Description | Default |
|---|---|---|---|
| ***controllerSize*** | integer | Size of controllers in pixels | 100 |
| ***controllerSizeAuto*** | boolean | If true, controller sizes will be automatically adapted to comp size, according to *Duik.settings.controllerSizeHint* | true |
| ***controllerSizeHint*** | integer | when controllerSizeAuto is true, 0 = small, 1 = medium, 2 = big | 1 |
| | | | |

# Duik.utils

Some useful methods.

## Methods

*Duik.utils.prepareProperty(property,isFX,index,depth,parentName)*

| Name | Description | Return |
|---|---|---|
| *prepareProperty(property, isFX, index, depth, parentName)* | Prepare property to be rigged | true if property can set expression, false otherwise |
| | | |
| | | |
| | | |

***Duik.prepareProperty(property,isFX,index,depth,parentName)***

Prepare the given property to be rigged.
*isFX*, *index*, *depth*, *parentName* will be filled by the method with the values corresponding to this property.

parameters:

property | Property
isFX | boolean
index | integer
depth | integer
parentName | string

returns

true if property can set expression, false otherwise