Справочное руководство SimpleUI Library for Adobe ExtendScript

Оглавление

Обновлено 20.08.2014

Общие сведения

Библиотека SimpleUI представляет набор средств, ориентированных на работу с пользовательским UI и подсистемой ScriptUI в сценариях JavaScript (ExtendScript) для приложений Adobe Creative Suite/Cloud (CS/CC).

Структурно, библиотека SimpleUI входит в общий репозиторий https://github.com/SlavaBuck/Includes, содержащий одновременно несколько библиотек общего назначения. Библиотека SimpleUI состоит из заголовочного файла `SimpleUI.jsx` и папки с файлами самой библиотеки `SimpleUI\`. Несмотря на размещение в общем репозитории, библиотека не имеет каких бы то ни было зависимостей от внешнего кода. Таким образом, для включения и использования библиотеки в целевом сценарии достаточно перенести указанные файл и папку в любое выбранное расположение в локальной файловой системе и включить заголовочный файл `SimpleUI.jsx` в пользовательском скрипте с помощью директивы #include.

В пользовательском файле сценария библиотека представлена отдельной областью имён с коротким наименование `SUI` и доступна через соответствующий глобальный объект модуля:

```
// включение библиотеки SimpleUI
#include “path/to/SimpleUI.jsx”

$.writeln(“Имя библиотеки: ”, SUI.name);
$.writeln(“Версия библиотеки: ”, SUI.version);
```

Архитектура и состав библиотеки

Библиотека SimpleUI имеет модульную архитектуру и включает несколько относительно независимых модулей:

- UIColors — именованная таблица стандартных html/x11 цветов и функции для работы с цветом в различных форматах;
- UIControls — коллекция расширенных ScriptUI-элементов управления (ProgressBar, ScrollablePanel, Separator, WebLink и др...);
- UIImage — расширение базового класса ScripUIImage (реализована поддержка масштабирования объекта ScripUIImage в пользовательских элементах управления), функции для конвертации ScripUIImage в ресурсные строки и обратно;
- ESTKLib — именованная коллекция стандартных графических ресурсов (пиктограмм) в ESTK;

Каждый модуль представлен собственным одноимённым заголовочным файлом, расположенном в общей библиотечной папке SimpleUI и может быть самостоятельно включён в целевом файле сценария, без необходимости включения всей библиотеки SimpleUI целиком.

Модули в рамках библиотеки SimpleUI представлены собственными одноимёнными пространствами имён. Однако, в целях сокращения имён ссылок, пространство имён каждого модуля расширяет общее пространство имён библиотеки SimpleUI и полностью дублируется объекте модуля SUI. Следующий пример демонстрирует сказанное:

```
#include “path/to/SimpleUI.jsx”

$.writeln(“Имя модуля: ”, SUI.UIControls.name); // пространство имён модуля
$.writeln(SUI.UIControls.initWindow === SUI.initWindow); // => true
```

Модуль UIColors

Заголовочный файл: `SimpleUI/UIColors.jsx`

Модуль содержит именованную таблицу стандартных html/x11 цветов, представленную объектом COLORS и ряд функций для работы с цветом.

Объекты и методы в составе пространства имён модуля UIColors:

```
toRGB(color /* uint */)
```

Конвертирует целое в формат rgb (Array[3] – массив из трёх целых чисел)

```
toRGBA(color /* uint */ [, alpha /* float 0..1.0 */])
```

Конвертирует целое в формат rgba (Array[4] – массив из четырёх чисел с плавающей точкой (от 0 до 1.0, стандартный формат представления цвета в ScriptUI). Может принимать второй необязательный параметр alpha — прозрачность. Выходной массив rgba дополняется четвёртым компонентом alpha (по умолчанию используется значение 1 — отсутствие прозрачности).

```
RGBtoValue(r, g, b /* 3 uint */);
RGBtoValue(rgb /* Array[3]  uint */);
```

Конвертирует RGB в целое значение. Принимает цвет в формате трёх целых значений от 0 до 255 (компоненты R, G, B) и возвращает целое значение (выполняет обратное функции toRGB() преобразование). Имеет альтернативную форму вызова и допускает передачу параметров в виде массива RGB (Array[3] – массив из трёх целых чисел).

```
RGBAtoValue(r, g, b [,alpha /* 3-4 float */]);
RGBAtoValue(rgba /* Array[3-4] float */);
```

Конверитрует RGBA в целое значение. Принимает цвет в формате трёх или четырёх чисел с плавающей запятой от 0 до 1.0 (компоненты R, G, B и A – альфа/прозрачность) и возвращает целое значение (выполняет обратное функции toRGBA() преобразование). Имеет альтернативную форму вызова и допускает передачу параметров в виде массива RGBA (Array[3-4] – массив из трёх или четырёх чисел с плавающей запятой). Четвёртый компонент alpha (альфа) в процессе преобразования игнорируется.

```
RGBtoRGBA(r, g, b [,alpha /* 4 uint */]);
RGBtoRGBA(rgb /* Array[3-4] uint, Array[4] = alpha 0..255 */]);
```

Конвертирует цвет, заданный в формате RGB в формат RGBA. Принимает параметры в виде 3 или 4 целых чисел (представляющих компоненты RGB и необязательно – компоненту alpha) или массива в формате RGB. Возвращает массив в формате RGBA. Если в параметрах передаётся 4 компонент alpha – параметр рассматривается как целое от 0 до 255, конвертируется и возвращается в массив RGBA, если параметр alpha не указан – в качестве 4-го параметра в массив RGBA возвращается 1.

```
RGBAtoRGB(r, g, b [,alpha /* 4 float */]);
RGBAtoRGB(rgb /* Array[3-4] float, Array[4] = alpha 0..1.0 */]);
```

Конвертирует цвет, заданный в формате RGBA в формат RGB. Принимает параметры в виде 3 или 4 чисел с плавающей запятой (представляющих компоненты RGBA) или массив в формате RGBA. Возвращает массив в формате RGB. Четвёртый компонент alpha (альфа) в процессе преобразования игнорируется.

```
parseColor(c /* uint||string(name of COLOR)||RGB-array||RGBA-array */)
```

Универсальная функция, возвращает строковое представление значения цвета в шестнадцатеричном формате (так называемый формат hexTriplet: "0xRRGGBB"). Аргумент анализируется на предмет соответствия одному из стандартов представления цвета и может собой представлять целое число, массив целых чисел или чисел с плавающей запятой, а также строчное наименование цвета, соответствующее одному из полей объекта COLORS. Пример:

```
$.writeln(parseColor("Red"));     // => '0xFF0000'  COLORS.Red == 0xFF0000
$.writeln(parseColor(16711680));  // => '0xFF0000'
$.writeln(parseColor([1, 0, 0])); // => '0xFF0000'
```

```
COLORS:{object}
COLORS.AliceBlue == 0xF0F8FF;    cAliceBlue == [.94, .97, 1, 1];
COLORS.AntiqueWhite == 0xFAEBD7; cAntiqueWhite == [.98, .92, .84, 1];
...
```

Стандартные html/x11 цвета (см. файл SimpleUI/colors.jsxinc, всего 142 значения). Дополнительно с объектом COLORS в глобальное пространство имён экспортируются одноимённые переменные, начинающиеся с малой буквы 'c' и представляющие соответствующие цвета в формате RGBA. Таким образом, в пользовательском сценарии можно использовать короткие записи вида:

```
w.graphics.backgroundColor = w,graphics.newBrush(0, cBrown);
```

вместо:

```
w.graphics.backgroundColor = w,graphics.newBrush(0, [.64, .16, .16, 1]);
```

Все объекты и методы, представленные в составе пространства имён модуля UIColors, экспортируются в глобальное пространство имён и доступны как глобальные объекты и функции:

```
#include "path/to/UIColors.jsx";

$.writeln(cRed);                // => 1,0,0,1
$.writeln(parseColor(cRed));    // => 0xFF0000
$.writeln(COLORS.Red);          // => 16711680
$.writeln(toRGBA(COLORS.Red));  // => 1,0,0,1
```

Модуль UIControls

Заголовочный файл: `SimpleUI/UIControls.jsx`

Модуль содержит коллекцию расширенных ScriptUI-элементов управления и ряд методов общего назначения:

- `FloatingProgressBar` — плавающее окно с прогрессбаром;
- `ScrollablePanel` — скроллируемая панель;
- `Separator` — разделительная линия;
- `UnitBox` — настраиваемое поле редактирования;
- `WebLink` — гиперссылка;

Объекты и методы в составе пространства имён модуля UIControls:

`initWindow(win /* Window */);`

Инициализация ScriptUI-объекта Window. Метод применяется к объекту диалога, переданного в аргументе `win`, сразу после его создания, выполняет рекурсивный обход и, в случае обнаружения, инициализирует все расширенные ScriptUI-элементы управления, которые будут обнаружены в объекте диалога. Дополнительно метод производит корректную инициализацию обработчиков масштабирования для диалогов, имеющих установленное свойство `resizeable:true`.

```
// инициализация диалога с установленным свойством resizeable:true -
// производится инициализация обработчиков onResize()/onResizing():

var w = new Window ("dialog { properties:{resizeable:true} }");
SUI.initWindow(w);
w.show();
```

```
// инициализация диалога, содержащего расширенные ScriptUI-элементы:

var w = new Window ("dialog {       \
    st0:StaticText { text:'Статический текст' }, \
    sp0:Panel { isSeparator:true }, \
    box:Group { isUnitBox:true }    \
");

// ниже вызов SUI.initWindow(w) заменяет необходимость 'поштучной'
// инициализации объектов Separator и UnitBox:
//     SUI.initSeparator(w.sp0);
//     SUI.initUnitBox(w.box);
SUI.initWindow(w);

w.show();
```

Метод может использоваться к объекту диалога многократно, например, после обновления свойств ориентации для диалога или контейнера и т.п.

`isContainer(control /* ScripUIobject||string */);`

Метод возвращает true, для всех ScriptUI элементов, имеющих контейнерный тип (один из panel, group, tabbedpanel, tab, dialog, palette, window). В качестве аргумента принимает либо строку с именем типа, либо указатель на графический элемент управления:

```
// w – объект диалога из предыдущего примера:

SUI.isContainer(w);       // => true
SUI.isContainer(w.st0);   // => false
SUI.isContainer('Panel'); // => true  (метод не чувствителен к регистру)
```

`Separator:{object}`

Вспомогательный объект, инкапсулирующий ресурсную строку UI-элемента Separator (см. Компонент Separator). Содержит единственное свойство:

`rcString:{string}` – ресурсная строка UI-элемента Separator

и метод `toString()`, возвращающий свойство `rcString`.

Может быть включён как слагаемое в составе длинных ресурсных строк диалоговых окон:

```
var w = new Window ("dialog {{   \
    st0:StaticText { text:'Статический текст' }, \
    sp0:" + SUI.Separator + "    \
    box:Group { isUnitBox:true } \
");
```

или как аргумент ScriptUI-метода `.add()`:

```
var w = new Window ("dialog");
w.add(SUI.Separator);
```

`UnitBox:{ object}`

Вспомогательный объект, инкапсулирующий ресурсную строку UI-элемента UnitBox (см. Компонент UnitBox). Дополнительно объект содержит несколько свойств, для настройки параметров по умолчанию UI-элемента UnitBox:

`rcString:{string}` – ресурсная строка UI-элемента UnitBox
`defaultType:'cm'` – тип поля редактирования (соответствует типам UnitValue)
`defaultCharacters:8` – начальный размер поля редактирования;

Может быть включён как слагаемое в составе длинных ресурсных строк диалоговых окон:

```
var w = new Window ("dialog {    \
    st0:StaticText { text:'Статический текст' }, \
    sp0:" + SUI.Separator + "    \
    box:" + SUI.UnitBox   + "    \
");
```

или как аргумент ScriptUI-метода `.add()`:

```
var w = new Window ("dialog");
w.add(SUI.UnitBox);
```

`WebLink:{object}`

Вспомогательный объект, инкапсулирующий ресурсную строку UI-элемента WebLink (см. Компонент WebLink). Дополнительно объект содержит несколько свойств, для настройки параметров по умолчанию UI-элемента UnitBox:

`rcString:{string}` – ресурсная строка UI-элемента UnitBox
`defBackgroundColor:{Array[4]}` – цвет родительского окна в формате RGBA-array;
`defWebLinkColor:{Array[4]}` – цвет ссылки (по умолчанию синий);

Может быть включён как слагаемое в составе длинных ресурсных строк диалоговых окон:

```
var w = new Window ("dialog {   \
    url: " + SUI.WebLink + "  \
");
```

или как аргумент ScriptUI-метода `.add()`:

```
var w = new Window ("dialog”);
w.add(SUI.WebLink);
```

Компонент FloatingProgressBar

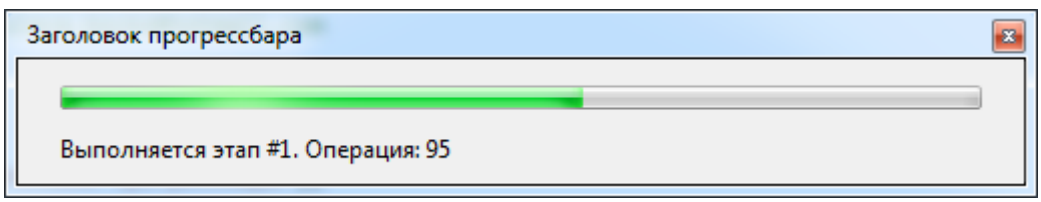
Компонент представляет собой плавающую панель с расположенным в ней индикатором индикатор выполнения задания — прогрессбаром.

Пример создания и использования прогрессбара:

```
var pBar = new SUI.FloatingProgressBar("Заголовок прогрессбара");

// Длинная операция #1
// методу hit() - передаётся строка, которая меняется на каждой итерации
цикла
// и выводится под линией прогрессбара:
var title = "Выполняется этап #1.";

pBar.reset(title, 100); // Прогрессбар рассчитывается на сто итераций

for(var i=0 ; i < 100; ++i, pBar.hit(title + " Операция: " + i) ) {
   $.sleep(10);
}

// Операция #2
var i;
title = "Выполняетсяе этап #2.";
pBar.reset(title, 10); // Прогрессбар переустанавливается на десять итераций
for( i=0 ; i < 10; ++i ) {
    // hit() без параметров еквивалентно pBar.step(i) - с каждым вызовом
hit() внутренний
    // счётчк прогрессбара увеличивается на +1
   pBar.hit();
   $.sleep(300);
}

pBar.close();
```

При создании, конструктору `SUI.FloatingProgressBar()` может быть передана строка, которая будет использована в качестве заголовка панели с прогрессбаром.

Следующий вызов `pBar.reset(title, 100);` настраивает прогрессбар на отработку 100 итераций. При этом аргумент `title` используется как подпись под полосой индикатора прогресбара:



После выполнения метода `.reset()`, панель с индикатором прогресса выводится на экран, а сам прогрессбар готов к выполнению итераций.

Итерации для прогрессбара выполняются с помощью вызова метода `.hit()`. Каждый вызов метода `.hit()` увеличивает значение индикатора на расчётную величину. Таким образом, метод `.hit()` вызванный `n`-ое количество раз, предварительно указанное при вызове метода `.reset(title, n)`, перемещает индикатор прогресса в положение 100%. При вызове, методу `.hit()` может передаваться строка, в примере: `pBar.hit(title + " Операция: " + i)`. Переданная строка располагается под индикатором прогресса, используется как подпись для текущей итерации и заменяет собой строку, указанную при вызове метода `pBar.reset(title, ..);`

После выполнения всех итераций прогрессбар следует закрыть, вызвав метод `pBar.close()`.

Компонент ScrollablePanel
// TODO:

Компонент Separator
// TODO:

Компонент UnitBox
// TODO:

Компонент WebLink
// TODO:

Модуль UIImage
// TODO:

Модуль ESTKLib
// TODO:

Дополнительные сведения
- Открытый общий репозиторий с библиотеками: https://github.com/SlavaBuck/Includes/;
- Репозиторий с библиотекой SimpleUI: https://github.com/SlavaBuck/Includes/tree/master/SimpleUI;
- Связаться с автором можно со страницы проекта либо по эл. почте slava.boyko@hotmail.com;

Украина, г. Киев
slava.boyko@hotmail.com