SYHUNT SANDCAT: EXTENSION DEVELOPMENT

The information in this document applies to **version 6.8** of Syhunt Hybrid.

THE SANDCAT EXTENSION SYSTEM

Syhunt Sandcat extensions are developed using a combination of **HTML**, **CSS** and **Lua** script. Sandcat Extensions, Tasks and Console commands can execute Lua code with all its standard functions, including file IO support. TIScript (a programming language derived from JavaScript) can also be used to extend and manipulate the Sandcat user interface.

Sandcat Extensions are deployed as ZIP files (renamed **.scx**) under [SandcatDir]\Packs\Extensions\

EXTENSION MANIFEST

During startup Sandcat will check all extensions for a Manifest.json file contained within the extension's ZIP package. The following is an example of a manifest file:

```
{
 "name": "Hello World Extension",
 "version": "1.0",
 "author": "Me",
 "script": {
  "filename": "hello.lua",
  "init": "Hello:register()",
 }
}
```

MANIFEST KEYS

- **name***: The extension name
- **version***: The extension version. If 'file:[dllfilename]' is used as version, the browser will read the version from the supplied DLL file.
- **description**: The extension description
- **script***: The Lua code
  - filename: The main Lua script file contained within the extension's ZIP package. This script will be loaded and executed during the browser startup
  - init: Can contain a Lua script to be executed during the browser startup. If script.filename has been

provided, executes after the script file has been loaded.
- shutdown: A Lua script to be executed during the browser shutdown

(*) = required keys

HELLOWORLD EXAMPLE

How to make a simple hello world extension:

- Create a folder somewhere on your system to contain your extension's files.
  1. Inside the folder, create Manifest.json with the following content:

```
{
 "name": "Hello World Extension",
 "version": "1.0",
 "author": "Me",
 "script": {
  "filename": "hello.lua",
  "init": "Hello:register()",
 }
}
```

- Create a Lua script called hello.lua with the following content:

```
Hello = {}

function Hello:showmsg()
 app.showmessage('Hello World!')
end

function Hello:register() -- will be called during startup
 local html = [[
 <div
 class="button"
 onclick="Hello:showmsg()"
 style="foreground-image: url(Hello.scx#world.png);"
 />
 ]]
 browser.navbar:inserthtml(1,'#toolbar',html)
end
```

- Place  icon file in the same folder
- Zip the files and change the extension to .scx. The zip file can be called Hello.scx.
- Copy the Hello.scx file to the **Packs\Extensions** folder, which is located in the root of the Sandcat

Browser folder.
- Launch the Sandcat Browser. Now in the navigation bar, you should see the globe icon. Now try clicking on it.

MORE EXAMPLES

- Syhunt Community Extensions

SECURITY CONSIDERATIONS WHEN DEVELOPING EXTENSIONS

Sandcat Extensions, built using HTML and Lua, can execute Lua code with file IO support. For this reason, you should not load a string as part of the HTML of an extension user interface without filtering or validating the content before.

This applies to the following functions:

- **app.showdialogx()**
- **app.showmessagex()**
- **browser.loadpagex()**
- **browser.newtabx()**
- **tab:loadx()**
- **[anyuizone]:loadx()**

NEVER USE THE EXTENSION USER INTERFACE FOR REGULAR BROWSING

You can use http for displaying media files, such as images, but do not try to load web sites using supported protocols. You will be loading an entirely new user interface, which as mentioned before has desktop application privileges.

If you need to load a web page, use the function **tab:gotourl()**, which will load the web page using the standard, safe for browsing Chromium navigator.

SECURITY BUG REPORTING

Security bugs should be reported directly to security@syhunt.com. Low risk security bugs can be reported by opening an issue here. If you are unsure about the risk level, please report it via email.

APP LIBRARY

Contains mostly application and window related functions

- **app.ask_yn** (msg [,caption]) : Asks a question. Returns true if the Yes button is pressed and false if No is pressed. The second parameter is optional. Returns `boolean`
- **app.gettitle** (): Gets the application window title. Returns `string`
- **app.openfile** ([filter, default_ext, default_filename]): Displays the open file dialog. Returns the entered filename or an empty string if the Cancel button has been pressed. All parameters are optional. Returns `string`
- **app.savefile** ([filter, default_ext, suggested_filename]): Displays the save file dialog. Returns the entered filename or an empty string if the Cancel button has been pressed. All parameters are optional. Returns `string`
- **app.selectdir** ([caption]): Displays the select directory dialog. Returns the selected directory or an empty string if the Cancel button has been pressed. All parameters are optional. Returns `string`
- **app.settitle** (s): Sets the application window title
- **app.showalert** (s): Displays a warning message to the user
- **app.showinputdialog** (prompt [, default_value, caption]): Displays a simple input dialog. Returns the entered string or the default value string if the Cancel button has been pressed.
- **app.showdialogx** (html [, id]): Displays a custom Sandcat extension dialog. The id parameter is optional, can be used to assign an unique identifier to the dialog.
- **app.showmessage** (s): Displays a message to the user
- **app.showmessagesmpl** (s): Displays a message to the user using a simple dialog
- **app.showmessagex** (html): Displays a HTML formatted message to the user.
- **app.update** (): Updates the screen; Ensures that the window is completely drawn.

VARIABLES

- **app.dir**: stores the name of the Sandcat application directory

- **app.datadir**: stores the name of the Sandcat application data directory

BROWSER LIBRARY

CORE BROWSER METHODS

- **browser.addlibinfo** (name, version, author [,luacode]): Adds information about a library to the About Sandcat screen. Should be called by extensions during startup

  - *version* - the library version. If 'file:[dllfilename]' is used as version, the browser will read the version from the DLL file.
  - *luacode* - A code to be executed when the library information icon is clicked. This parameter is optional.

- **browser.bookmark** ([title,url]): Adds a page to the bookmarks. This function bookmarks the page of the active tab if no parameters are provided

- **browser.cleardata** (type): Clears browsing data. The type parameter can be: cache, appcache, cookies, databases, history, settings

- **browser.closepage** (pagename): Closes an extension page by its name

- **browser.closetab** (tabname): Closes a tab by its name. If no param is supplied, closes the active tab

- **browser.closetabs** (except): Closes all tabs. If a tab name is supplied as parameter, closes all tabs except the specified tab. E.g. browser.closetabs(tab.name) will close all but active tab

- **browser.dostring** (luacode): Runs a Lua script

- **browser.exit** (): Closes the browser

- **browser.getpackfile** (pakfilename, filename): Gets the contents of a file that is inside an extension package. Returns `string`

- **browser.gettaskinfo** (taskid): Gets details about a Sandcat Task, launched using the tab:runtask() function. Returns `table`

- **browser.gototab** (tabname): Goes to the specified tab

- **browser.loadpagex** (luatable): Loads an extension page in a separate page.

  - *name* - page name
  - *html* - page contents
  - *table* - the name of a Lua table. Associates the HTML elements of the page with the table. See UI Manipulation for details.
  - *noreload* - avoids reloading the page if loadpagex() is called again with the same table parameters

- **browser.newtab** (url [,source]): Opens an URL in a new tab. The source parameter is optional - if specified, the page will be loaded from it. Returns the tab number. Returns `integer`

- **browser.newtabx** (luatable): Creates a custom extension tab. Returns the tab number. Returns `integer` The following keys can be provided (all of them optional):

  - *activepage* - the active tab page (ex: browser, source, log...)
  - *icon* - an icon url
  - *html* - page contents
  - *shownavbar* - if false, hides the navigation bar.
  - *table* - the name of a Lua table. Associates the HTML elements of the page with the table. See UI Manipulation for details.
  - *tag* - an unique string to identify the tab. If newtabx() is called again and a tab with the supplied tag has been already created, Sandcat goes to the tab instead of creating a new one. Additionaly, if a *loadnew* key is supplied and is true, the tab page is reloaded from the HTML string supplied as the second parameter.
  - *title* - the tab title
  - *toolbar* - a custom toolbar source (eg: MyExtension.scx#MyToolbar.html)

- **browser.newwindow** ([url]): Opens a new browser window. If a URL is provided as parameter, opens the URL

- **browser.removetask** (taskid): Stops and removes a Sandcat Task from the tasks list

- **browser.setactivepage** (pagename): Sets the active page by its name (eg, browser, source, log, etc).

- **browser.setinitmode** (modename , luacode): Sets a Lua code to be executed when the Sandcat executable is launched with the mode:modename parameter. This allows a Sandcat extension to customize the browser user interface during startup.

- **browser.setsearcheng** (name, queryurl, iconurl): Sets a new search engine

- **browser.showreqbuilder** (): Loads the Request Builder bar

- **browser.showurl** (url [,source]): Loads an URL in the bottom bar. The source parameter is optional - if supplied, the page will be loaded from it

- **browser.stoptask** (taskid [,reason]): Stops a Sandcat Task. The second parameter is optional, can be used to explain the reason of the stop

- **browser.suspendtask** (taskid [,resume]): Suspends a Sandcat Task. The second parameter is optional - if supplied and true, resumes the task.

META TABLES

BROWSER.OPTIONS

Allows to set the visibility of special UI parts

name	type	description
showbottombar	boolean	Sets the visibility of the bottom bar
showconsole	boolean	Sets the visibility of the Sandcat Console
showheaders	boolean	Sets the visibility of the live headers
showpagestrip	boolean	Sets the visibility of the page strip
showsidebar	boolean	Sets the visibility of the sidebar

BROWSER.INFO

Returns various details about the browser

- abouturl
- cachedir
- configdir
- commands
- downloads
- errorlog
- exefilename
- extensions
- iconfilename

- libraries
- name
- options
- proxy
- tasks
- useragent
- version

BROWSER.JSVALUES

Stores temporary JSON values

CATARINKA

Catarinka is a multi-purpose set of Lua extensions developed to be used in the Sandcat Browser. Currently, this library extends Lua with over 60 functions and some useful classes.

In Sandcat, Catarinka is available as the **ctk** global. For a list of available functions, see functions.md.

CLASSES

All Catarinka classes (described in `classes.*` under the Catarinka Docs are available.

Each class has a "new" method that must be used for creating the object and a "release" method for freeing it.

CONSOLE LIBRARY

Sandcat Browser comes with Sandcat Console, a command console with several useful commands and extension possibilities.

FUNCTIONS

- **console.addcmd** (name,luacode,description) : Adds a custom command to the Sandcat Console

- **console.clear** (): Clears the contents of the Sandcat Console

- **console.gethandler** (): Gets the current command handler. Returns `string`

- **console.reset** (): Restores user settings, default handler and clears the console.
- **console.setcolor** (htmlcolor) : Sets the background color
- **console.setfontcolor** (htmlcolor) : Sets the font color
- **console.sethandler** (cmdname): Redirects all future commands to the supplied cmd name
- **console.write** (v) : Writes the parameter to the Sandcat Console without creating a new line
- **console.writeln** (v): Writes a new line to the Sandcat Console

CMD OBJECT

The cmd object allows you to get the parameters of a Sandcat Console command. The object is automatically created when Sandcat starts. Its available methods and properties are described below.

name	type	description
cmd.name	string	Gets the name of the last command
cmd.params	string	Gets the parameters of the last command

ADDING COMMANDS

Below you can find how to extend the console with custom commands. Currently, there are two methods for adding new commands.

METHOD 1: AS PART OF A SANDCAT EXTENSION

New commands can be added using the Lua language via the **console.addcmd()** function during the Sandcat extension initialization or at any moment after the initialization. The first parameter must contain the command name and its arguments (if any), the second parameter the Lua code to be executed, and the third parameter must contain a simple description of the command. See below a few examples.

```
MyCommands = {}

function MyCommands:Google(query)
 if query ~= '' then
  browser.newtab('https://www.google.com/search?q='..query)
 end
end

function MyExtension:init()
 -- Google Search command
 console.addcmd('search [query]','MyCommands:Google(cmd.params)','Searches Google')
 -- Simple print command
 console.addcmd('say [str]','print(cmd.params)','Prints a string of text')
end
```

METHOD 2: AS AN EXTERNAL COMMAND FILE

New commands can also be added by creating a Lua script file which must be placed in the Scripts/Commands directory. The first line of the script must be a comment containing the command name and its arguments (if any). The second line should contain a simple description of the command. See below an example of a command file.

say.lua:

```
-- say [str]
-- Prints a string of text
if cmd.params ~= '' then
 print(cmd.params)
end
```

PREFS LIBRARY

Can be used to read or change Sandcat preferences

- **prefs.get** (key, novalue): Gets the value of a browser setting. If key is not found, returns the value supplied in the second parameter; otherwise, returns nil. Returns `variant`
- **prefs.getdefault** (key): Gets the default value of a browser setting. Returns `variant`
- **prefs.getall( )**: Returns the value of all settings as JSON string. Returns `json string`
- **prefs.getalldefault** (): Returns the default value of all settings as JSON string. Returns `json string`

- **prefs.load** (json): Loads the browser settings from a JSON string
- **prefs.regdefault** (key, value): Registers the default value of a browser setting.
- **prefs.save** (): Saves the browser settings to the disk.
- **prefs.set** (key, value): Sets the value of a browser setting.
- **prefs.update** (): Notifies all open tabs about a setting or multiple settings update

EXTENSIONPACK OBJECT

Reads Sandcat Extension (.scx) pack files

CREATION

- **expansionpack:new** (): Creates and returns a new extension pack object.

METHODS

- **dofile** (scriptfilename): Loads and runs a Lua script that is inside the extension package
- **fileexists** (textfilename): Returns true if a file is present inside the extension package. Otherwise, returns false. Returns `boolean`
- **getfile** (textfilename): Gets the contents of a text file that is inside the extension package. Returns `string`
- **require** (modulename): Loads and returns the Lua module that is inside the extension package. Similar to Lua's require().
- **release** (): Frees the extension pack object

PROPERTIES

name	type	description
filename	string	Gets or sets the extension pack filename (eg: MyPack.scx)

OSR OBJECT

Creates Off-Screen Chromium Renderer processes

CREATION

- **osr:new** () Creates and returns a new Sandcat Chromium OSR object

EXAMPLE

```
local cef = osr:new()
cef.onsetsource = function(s) app.showmessage(s) end
cef.onrequestdone = function(t) app.showmessage(t.url) end
cef:loadurl('http://www.syhunt.com/sandcat/')
```

METHODS

- **loadrequest** (method, url[,postdata]): Sends a HTTP request and loads its response. The third parameter is optional; can be used in POST requests. This method can alternatively be called with a single table parameter (see below)

- **loadrequest** (luatable): Sends a custom HTTP request and loads its response. Accepts the same keys as the tab:loadrequest() method.

- **loadsource** (source, url): Renders a source code string

- **loadurl** (url): Goes to the supplied url

- **reload** (ignorecache): Reloads the page. If the first parameter is supplied and is true, the cache will be ignored

- **runjs** (jscode [,url, startline]): Executes a JavaScript code in the loaded page. The last two parameters are optional

- **runjs** (luatable): Performs a custom JavaScript call in the loaded page. Accepts the same keys as the tab:runjs() method.

- **sendrequest** (luatable): Sends a custom HTTP request. See Request for details. Accepts the same keys as the tab:loadrequest() method.

- **showauthdialog** ([username,password]): Displays the basic authentication dialog. The username and password parameters are optional. If supplied, they are used as default values in the dialog

- **stop** (): Stops loading a page

- **release** (): Frees the OSR object

PROPERTIES

name	type	description
captureurls	boolean	Enables or disables the capture of URLs during operation (Default: false)
downloadfiles	boolean	Enables or disables the download of files during operation (Default: false)
getsourceastext	boolean	Defines if the source must be returned as text
reslist	string	Gets the list of resources (images, CSS files, etc) of the loaded page
url	string	Gets or sets the URL of the renderer
urllist	string	Gets a list of URLs captured during operation (requires captureurls enabled)

EVENTS

- **onaddresschange**: If set, called when the URL of the renderer changes (eg, after a 302 redirect). Receives a string containing the new URL

- **onbeforepopup**:If set, called before creating a popup window Receives a table parameter containing the keys: url

- **onconsolemessage**: If set, called when there is a JavaScript execution error or when console.log() is called. Receives a table parameter containing the keys: message, source and line

- **ondownloadstart**: If set, called before starting a file download Receives a table parameter containing the keys: id, suggestedname

- **onloadend**: If set, called when the page finishes loading Receives a table parameter containing the keys: statuscode

- **onloaderror**: If set, called when there is a error during operations Receives a table parameter containing the keys: errorcode, errortext and failedurl

- **onloadstatechange**: If set, called when the loading state of a page changes. Receives a table parameter containing the keys: isloading, cangoback and cangoforward

- **onrequestdone**: If set, called after every request Receives a table as first parameter and a JSON string as second parameter. They contain request details in the form of keys

- **onresourcefound**: If set, called when resouce files (images, CSS, etc) are loaded. Receives a string containing the resource URL

- **onsetsource**: If set, called when the page finishes loading Receives a string containing the page source code

TAB OBJECT

The tab object is automatically created when Sandcat starts and is always connected to the active tab.

METHODS

- **tab:clearheaders** (): Clears the live headers

- **tab:clearlog** (): Clears the contents of the Log tab

- **tab:goback** (): Goes back in history

- **tab:goforward** (): Goes forward in history

- **tab:gotosrcline** (int): Goes to the source page and scrolls to a given line number

- **tab:gotourl** (url [, source]): Goes to the supplied url. The source parameter is optional - if supplied, the page will be loaded from it.

- **tab:loadrequest** (method, url [,postdata]): Sends a HTTP request, loads its response and adds the request to the Live Headers list. The third parameter is optional; can be used in POST requests. This method can alternatively be called with a single table parameter (see below).

- **tab:loadrequest** (luatable): Sends a custom HTTP request, loads its response and adds the request to the Live Headers list. The following keys can be provided:

    - method - the HTTP method (GET, POST, etc). Default GET url - the request URL (if not supplied, the tab URL will be used as URL)
    - postdata - the request data
    - headers - custom request headers

- ignorecache - If supplied, and is false, may load the page from the cache. Default true.
- useauth - If supplied, and is true, uses cached authentication credentials. Default true.
- usecookies - If supplied, and is false, ignores cookies. Default true. details - a short description for the request

- **tab:loadsourcefile** (filename): Loads the contents of source page from a file.

- **tab:loadx** (html [,tablename]): Loads an extension page. The second parameter is optional - if supplied, associates the HTML elements of the page with a Lua table. See UI Manipulation for details.

- **tab:log** (s): Prints a line to the Log page.

- **tab:logerror** (componenttitle, linenumber, msg): Manually adds an error item to the Script Errors screen.

- **tab:logrequest** (json [,responsetext]): Manually logs a request from a custom HTTP client class to the Live Headers. The second parameter is optional and can contain the response text. The following JSON keys can be provided:

  - method (required): the request method
  - url (required): the request URL. In case this is a low level request you can supply a host and a port key instead of an URL key.
  - postdata: the request data
  - headers: the request headers
  - responseheaders: the response headers
  - responsefilename: the name of a temporary file that must contain the response text or stream. The file will be automatically deleted after its contents are added to the Live Headers cache.
  - status: if not supplied, the status code will be read from the response headers
  - details: a short description for the request

- **tab:reload** (ignorecache): Reloads the page. If the first parameter is supplied and is true, the cache will be ignored.

- **tab:runjs** (jscode [,url, startline]): Executes a JavaScript code in the loaded page. The last two parameters are optional.

- **tab:runjs** (luatable): Performs a custom JavaScript call in the loaded page. The following keys can be provided:

  - code (required): the JavaScript code

- url: the JS URL
- startln: the start line number (default is 0)
- silent: If supplied, and is true, JS execution errors are not reported. Default false.

- **tab:runluaonlog** (msg,luacode): Sets a Lua script to be executed after a certain message is received through the JS console.log() method. This method should be used with caution.

- **tab:runtask** (luacode [,json, menuhtml]): Executes a Lua code in an isolated process. The second parameter is optional and can be used to pass parameters to the task process. Returns an unique task ID. Returns `string`

- **tab:search** (s): Performs a search using the active search engine

- **tab:sendrequest** (method, url [,postdata]): Sends a HTTP request and adds the request to the Live Headers list. The third parameter is optional. This method can alternatively be called with a single table parameter as explained below.

- **tab:sendrequest** (luatable): Sends a custom HTTP request and adds the request to the Live Headers list. Supports the same table keys as the tab:loadrequest() method (see above).

- **tab:showauthdialog** ([username, password]): Displays the basic authentication dialog. The username and password parameters are optional. If supplied, they are used as default values in the dialog.

- **tab:stopload** (): Stops loading a page.

- **tab:viewsource** (): Views the page source in an external editor.

PROPERTIES

name	type	description
capture	boolean	Enables or disables the live headers capture
capturebrowser	boolean	Enables or disables the live headers capture just for browser requests
downloadfiles	boolean	Enables or disables the download capability
headersfilter	string	Gets or sets the live headers filter
icon	string	Gets or sets the tab icon

name	type	description
lastjslogmsg	string	Gets the last message from a JS console.log() call
loadend	string	Sets a Lua script to be executed after the page finishes loading
loadendjs	string	Sets a JavaScript to be executed after the page finishes loading
logtext	string	Gets or sets the contents of the Log page
name	string	Gets the unique ID of the tab
reslist	string	Gets the list of resources (images, CSS files, etc) of the loaded page
screenshot	string	Takes a screenshot of the loaded page and returns its temporary filename
source	string	Gets or sets the source of the loaded page
status	string	Gets or sets the status bar text
title	string	Gets or sets the tab title
url	string	Gets the URL of the loaded page
zoomlevel	double	Gets or sets the zoom level

TASK OBJECT

Sandcat can launch Lua scripts to perform operations in an isolated process. The tasks can be monitored and managed from the Tasks page.

METHODS

Methods available from within an isolated process of a Sandcat Task:

- **task:browserdostring** (luacode): Runs a Lua script in the browser process.
- **task:dopackfile** (pakfilename, scriptfilename): Loads and runs a Lua script that is inside an extension package
- **task:logrequest** (json) Can be used to manually log a request from a custom HTTP client class to the Live Headers list. Accepts the same keys as the tab:logrequest() method.

- **task:getpackfile** (pakfilename, filename): Gets the contents of a file that is inside the specified extension package. Returns `string`

- **task:sendrequest** (json): Performs a custom HTTP request in the tab that launched the task and adds the request to the Live Headers list. Accepts the same keys as the tab:loadrequest() method, but here they must be provided as a JSON object.

- **task:setprogress** (position [, max]): Updates the progress bar from the task progress monitor panel. If the second parameter is omitted, 100 is used as maximum value

- **task:setscript** (event, luacode): Sets a Lua code to be executed during specific task progress monitor panel events (eg: onclick, ondblclick, onstop)

- **task:showmessage** (s): Displays a message to the user

- **task:stop** (reason): Manually stops the task. Sandcat will automatically call this method at the end of the task execution (if the method has not been called before).

PROPERTIES

name	type	description
caption	string	Gets or sets the task caption
id	string	Gets the unique ID of the task
pid	string	Gets the process ID of the task
status	string	Gets or sets the task status text

ADDITIONAL FUNCTIONS

In addition to the functions above, all functions above and all functions from the Catarinka library are also available from within a Sandcat Task.

- **getappdir** (): Returns the Sandcat installation directory. Returns `string`
- **parambool** (key, defaultvalue): Returns the value of a parameter key as a boolean value. If the supplied key is not found and a default value has been supplied, returns the default value. Returns `boolean`
- **paramint** (key, defaultvalue): Returns the value of a parameter key as an integer value. If the supplied

key is not found and a default value has been supplied, returns the default value. Returns `integer`

- **paramstr** (key, defaultvalue): Returns the value of a parameter key as a string. If the supplied key is not found and a default value has been supplied, returns the default value. Returns `string`
- **print** (v): Prints a line to the task log
- **printfailure** (v): Prints a line to the task log and paints the task progress monitor panel in red color
- **printfatalerror** (v): Prints a line to the task log and paints the task progress monitor panel in yellow color
- **printsuccess** (v): Prints a line to the task log and paints the task progress monitor panel in green color

SANDCAT UI MANIPULATION

METHOD 1: USING A LUA OR TISCRIPT SCRIPT TAG

Sandcat can associate the HTML elements of a Sandcat extension user interface with a Lua table. In order to enable this, just add a `SandcatUIX` meta tag to the extension UI code, as exemplified below.

```
<meta name="SandcatUIX" content="MyExtension">
<script type="text/lua">
function MyExtension:changeit()
  self.plaintext1.value = 'New text!'
end
</script>
<plaintext id="plaintext1"></plaintext>
<button onclick="MyExtension:changeit()">Demo</button>
<button onclick="MyExtension.plaintext1.value = 'Another value'">Demo 2</button>
```

As seen above, elements that have an id attribute can be manipulated via **[tablename].[yourelementid]**, which returns a Sandcat UI element object.

Alternatively, you can use a TIScript script tag:

```
<script type="text/tiscript">
$(#btndemo).onControlEvent = function(evt) {
 if (evt.type == Event.BUTTON_CLICK ) {
  $("#myplaintext").value = "New text!";
  }
}
</script>
<plaintext id="myplaintext"></plaintext>
<button #btndemo>Demo</button>
```

METHOD 2: USING LUA

You can associate the HTML elements of a Sandcat extension user interface with a Lua table by supplying a table name when calling the browser.loadpagex() method, as exemplified below.

interface.html:

```
<plaintext id="myplaintext"></plaintext>
<button onclick="MyExtension:changeit()">Demo</button>
```

interface.lua:

```
MyExtension = extensionpack:new()
MyExtension.filename = 'Demo.scx'

function MyExtension:show()
  local html = self:getfile('interface.html')
  browser.loadpagex({name='My Extension',html=html,table='MyExtension'})
end

function MyExtension:changeit()
  self.myplaintext.value = 'New text!'
end
```

METHOD 3: USING TISCRIPT AND THE UI ZONE'S EVAL() METHOD

interface.html:

```
<plaintext id="myplaintext"></plaintext>
<button onclick="MyExtension:changeit()">Demo</button>
```

interface.lua:

```
MyExtension = extensionpack:new()
MyExtension.filename = 'Demo.scx'
MyExtension.zone = browser.pagex

function MyExtension:show()
  local html = self:getfile('interface.html')
  browser.loadpagex({name='My Extension',html=html,table='MyExtension'})
end

function MyExtension:changeit()
  self.zone:eval('$("#myplaintext").value = "New text!"')
end
```

UI ZONE OBJECT

Used to extend the Sandcat user interface

AVAILABLE UI ZONES

name	description
browser.navbar	Navigation bar
browser.pagebar	Page tab strip
browser.statbar	Status bar
browser.tabbar	Tab bar
tab.engine	Main engine of the active tab
tab.toolbar	Custom tab toolbar
reqbuilder.toolbar	Request Builder toolbar
dlg.prefs	Preferences dialog
dlg.about	About Sandcat dialog

UI ZONE METHODS

- **addhtml** (target, html): Extends the UI zone with custom HTML elements. This method should be called by extensions during startup

    - target: must be a CSS selector string
    - html: the HTML string you want to add

- **addhtmlfile** (target, html): Extends the UI zone with custom HTML elements from a file. This method should be called by extensions during startup

    - target: must be a CSS selector string
    - htmlfilename: the name of the HTML file you want to add

- **addtiscript** (tiscode): Extends the UI zone with custom TIScript. This method should be called by extensions during startup

- **eval** (tiscode): Evaluates a TIScript code. Returns `variant`

- **inserthtml** (index, target, html): Extends the UI zone with custom HTML elements. This method should be called by extensions during startup

    - index: insertion position (zero-based).
    - target: must be a CSS selector string
    - html: the HTML string you want to insert

- **inserthtmlfile** (index, target, htmlfilename): Extends the UI zone with custom HTML from a file. This method should be called by extensions during startup

- **loadx** (html [, tablename]): Loads the UI zone interface from a string. The second parameter is optional. If supplied, associates the HTML elements of the page with a Lua table. See UI Manipulation for details.

UI ELEMENT OBJECT

Used to manipulate the Sandcat user interface

METHODS

- **getattrib** (s): Gets the value of an attribute. Returns `variant`
- **setattrib** (s, v): Sets the value of an attribute
- **getstyle** (s): Gets the value of a style attribute. Returns `variant`
- **setstyle** (s , v): Sets the value of a style attribute

PROPERTIES

name	type	description
value	variant	Gets or sets the value of the element

TISCRIPT

TIScript is the scripting engine used by Sandcat for some of its user interface operations. TIScript uses JavaScript as a base with some Python features added: classes and namespaces, properties, decorators, etc.

More details can be found at: http://code.google.com/p/tiscript/wiki/LanguageAndRuntime

CUSTOM TISCRIPT FUNCTIONS AVAILABLE IN SANDCAT

Sandcat TIS Object:

- **Sandcat.Debug** (s): Outputs a debug string
- **Sandcat.GoToURL** (url [, newtab]): Goes to the supplied URL. The second parameter is optional. If supplied and is true, opens the URL in a new tab
- **Sandcat.PrefsSet** (key, value): Sets the value of a browser setting
- **Sandcat.ShowMessage** (s): Shows a message to the user
- **Sandcat.RunLua** (luacode): Runs a Lua script
- **Sandcat.Write** (s): Writes the parameter to the Sandcat Console without creating a new line.
- **Sandcat.WriteLn** (s): Writes a new line to the Sandcat Console
- **Sandcat.WriteValue** (key , value): Writes a value that can be later read using the browser.jsvalues Lua table