



Documentation

for the

Global

userChrome Button

Table of Contents

Lineage:.....	2
Menu:.....	3
Copy Button Name.....	4
Save this Button.....	4
Button's Attributes.....	4
Button Help.....	4
Global Objects, Handlers, and Functions.....	5
Global Clipboard Object.....	5
Mouse Click Event Handler.....	6
Create Message Function.....	7
Create Debug Function.....	8
Change History:.....	10

Lineage:

The original concept and code were authored by **LouCypher** as userChrome.js. This was then published as a button by **deepakjoshi04**. This launch was the result of the efforts of **SCClockDr & cblover**. The View Custom Button Codes Update is the work of **LouCypher**. If someone could add to this lineage please post or PM your info to “[Custom Buttons Forum Index -> 'Custom Button' Buttons -> \[ENHANCED\] userChrome, Add'l CB Menu & Global Routines](#)” thread.

Documentation for the Global userChrome Button

This button is only known to work with Firefox 2 and Custom Button extension 0.0.2 alpha on Windows.

Most of the code for this button was provided by different Custom Button forum developers at <http://custombuttons.phpbbnow.com/>, including among others:

deepakjosh04, a CB Regular, who joined the group 14 August 2006 and posted the initial button code 5 September 2006, <http://custombuttons.phpbbnow.com/viewtopic.php?t=235>. Also, see <http://custombuttons.phpbbnow.com/viewtopic.php?p=2357> for further commentary.

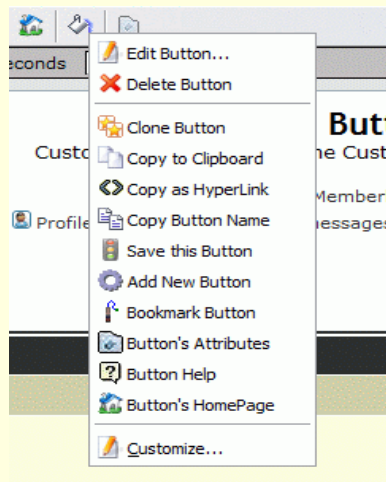
SCClockDr, who joined the group 25 December 2006 and is located in SC UpState.

cblover, a CB Regular, who joined the group 26 October 2006.

Menu:

When installed this button gives the user the following context menu options for all Custom Buttons:

1. Edit Button
2. Delete Button
3. Clone Button
4. Copy to Clipboard
5. Copy as HyperLink
- 6. Copy Button Name**
- 7. Save this Button**
8. Add New Button
9. Bookmark Button
- 10. Button's Attributes**
- 11. Button Help**
12. Button's HomePage
13. Customize



Context menu options 6, 7, 10, and 11 are new to this version of the button.

Also, it has the following additional properties:

1. A View Custom Button Codes option for the Firefox main context menu which allows one to view the code of a button by right clicking on its HyperLink.
 - **Version 20070328.00** – The Code fields now displayed in separate frames, with the Code field above the Initialization Code field.
2. Enables multiple Custom Button edit windows.
 - **Version 20070328.00** – Multiple edits of the same button now prevented. When you attempt to edit a button already being edited it will focus the button's edit dialog and not open a second dialog.

Additionally, it creates the following **new** global objects, handlers, and functions and makes them available to all button developers:

- 1. Global Clipboard Object.**
- 2. Mouse Click Event Handler.**
- 3. Message Object Function.**
- 4. Debug Object Function.**

Copy Button Name

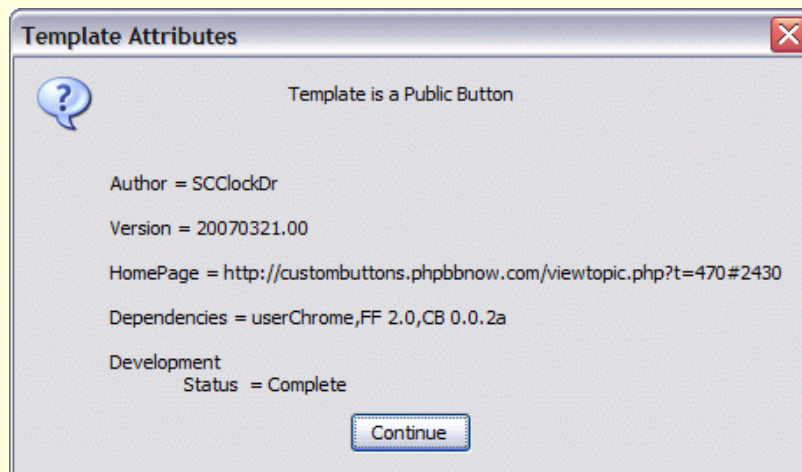
The Copy Button Name option places the Button's *Label*, *Public Attribute*, and *Version Attribute* onto the Windows System Clipboard.

Save this Button

The Save this Button option creates an HTML page containing the button's HyperLink information and various *Attributes*.

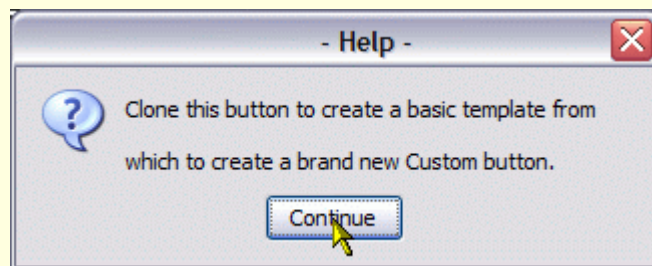
Button's Attributes

The Button's Attributes option displays a dialog containing the button's *Attributes* information and, also, places the information on the Windows System Clipboard.



Button Help

The Button Help option displays a dialog containing the Button's help *Attribute* information.



Global Objects, Handlers, and Functions

Global Clipboard Object

Property

sRead is an array container to house internal clipboard like data.

Methods

1. **gClipboard.write(*str*)** stuffs the system clipboard with the argument *str*
2. **gClipboard.read** returns the contents of the system clipboard
3. **gClipboard.clear** stuffs the system clipboard with ""
4. **gClipboard.Write(*str*)** stuffs the internal clipboard with the argument *str*
5. **gClipboard.Read** returns the contents of the internal clipboard

This Object can be instanced but there remains one and only one system clipboard; *var myClipboard = new Object(gClipboard);* is an example of this object being instanced.

Mouse Click Event Handler

gQuot = function(*evt*, *cButton*)

Args: *evt* - Mouse click event used to direct code execution based on the button clicked and any extra state keys held during the event.

cButton - Object of the calling button used to appropriately locate the context menu.

Returns: Nothing.

How this works: On a click the calling button is strobed via the *this.setAttribute("onclick", "gQuot(event, this);");* statement in the button's Initialization Code section. This passes the event and button object (*this*) to the event handler.

Execution:

1. First it checks for a shift + right click if that was the event, it calls *gShowPopup(cButton)* passing it the button object. The menu is handled in the usual way.
2. If not a shift + right click it proceeds to check for a click event function presence in the calling button.
 - a) Testing progresses down the tree till it arrives at a function with no extra key qualifier.
 - b) The first found function is called within the calling button. It passes the event object to the called function to provide the button author the info for further processing.
3. If no matches are found then *gShowPopup(cButton)* is called passing it the button object.

Calls:

Ctrl+LeftClick example:

1. *cButton.cleftclick(evt)* calls the button object function *cleftclick*. It passes the click event object to the button and expects nothing in return.
2. When control returns to *gQuot* function, program flow will encounter a break and the function exits to the button's calling attribute "onclick" statement.
3. Handling of the event is now complete.

Create Message Function

createMsg = function(title)

Args: *title* - Optional

Returns: New Message Object with title property initialized if provided.

How this works: `createMsg` uses the Object constructor method to create an object `Msg`

Properties:

1. `prompts`
2. `check`
3. `sTitle`

Methods: `Msg.aMsg(str, title)`

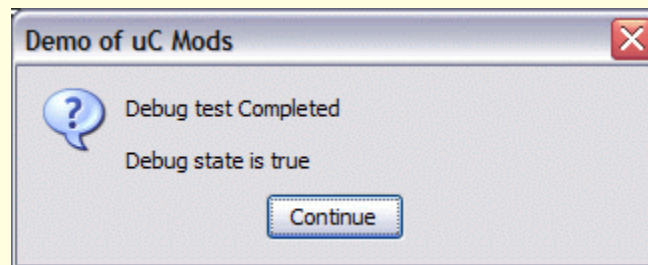
1. arg - *str* is the passed message string
2. arg - *title* is the optional dialog title

Calls: `this.prompts.confirmEx()`;

Setup: `MyObj = createMsg(['title'])`;

Use: `MyObj.aMsg('a message here');` // Display message

Example :



Create Debug Function

createDebug = function(*btn*)

Args: *btn* - Button Object

Returns: New Message Object for Debugging.

How this works: *createDebug* uses the Object constructor method to create an object *gMsg*

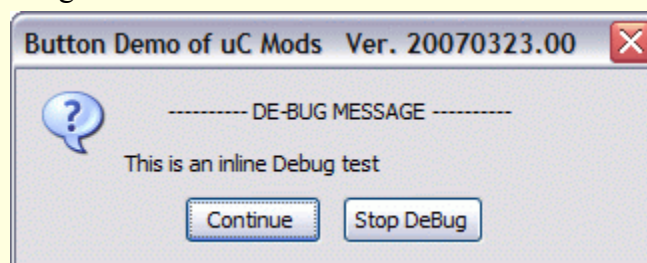
Properties:

1. *prompts* : Mozilla container for confirmex service
2. *strs* : String Array()
3. *sdebug* : Debug Title - '----- DE-BUG MESSAGE -----\n\n',
4. *swarn* : Warning Title - '+++++ WARNING +++++\n\n',
5. *strerr* : Error Array
6. *sVer* : Holds Button version Attribute data.
7. *check* : Object with value property : boolean. Used by Mozilla service.
8. *bugon* : Object with value property : boolean. True = DeBug is on.
9. *bugonlp* : Object with value property : boolean. True = DeBug loop is on.
10. *Func* : Holds function name if set.
11. *Name* : Holds Button label if set.

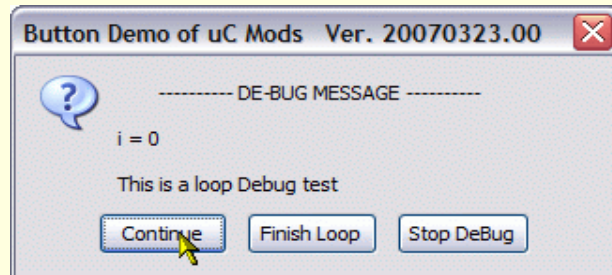
Methods:

1. *finit* : function(*btn*)
arg - *btn* is the passed button Object
task - Initialize the object with Button version & label data
returns - Nothing
2. *setErr* : function (*estr*)

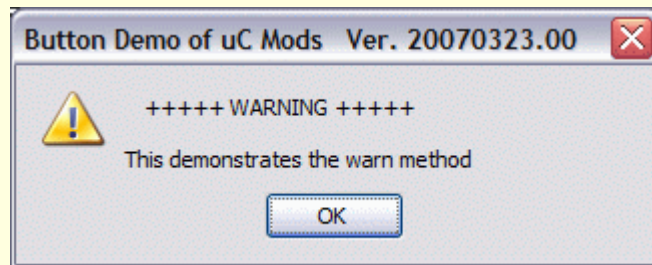
arg - *estr* is the passed message string
task - push new error string into the error string array
returns - index number to the passed error string
3. *bug* : function (*str*)
arg - *str* is the passed message string
task - Display Debug dialog with the passed message.
Forces *bugonlp* to true so next loop will be debugged.
Toggle off debug mode if requested by operator.
returns - Nothing



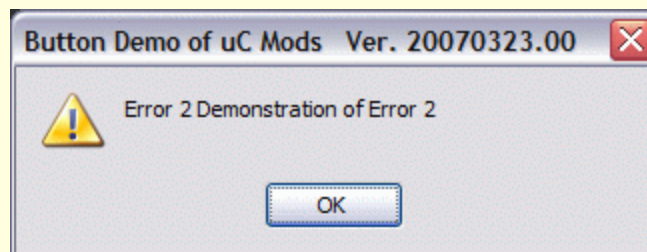
4. `bugloop` : function (*str*)
arg - *str* is the passed message string
task - Display loop Debug dialog with the passed message.
Requires *bugon* & *bugonlp* to be true for the dialog to display.
Toggle off either debug mode or debug loop mode if requested by operator.
returns – Nothing



5. `warn` : function (*str*)
arg - *str* is the passed message string
task - Display Warning dialog with the passed message.
Can be used to warn the operator of conditions
you the programmer wish to inform the operator about.
Can be used with try/catch to display caught errors.
returns – Nothing



6. `err` : function (*nerr*)
arg - *nerr* is the passed error message number
task - Display Error dialog with the numbered error message.
Can be used to warn the operator of conditions
you the programmer wish to inform the operator about
using a numbered error system.
returns - Nothing



Calls: *this.prompts.confirmEx()*;

Setup & Use:

```
de = createDebug(btn);  
de.bug('My test');  
loop {de.bugloop('My Loop Test');}
```

The following if Warning stuff wanted:

```
de.warn('Some Warning');
```

The following if Error stuff wanted:

```
var errNum = de.seterr('error string') // repeat as necessary  
if (some error) {de.err(4)} // would display error # 4 message
```

Change History:

- March 23, 2007 11:21:27 PM (EDT)
Minor change to the version # I failed to update prior to launch. Does not affect operation.

- March 28, 2007 00:20:00 AM (EDT) – Version 20070328.00
It has come to our attention that we omitted LouCypher from our credits for this button. **Lou, please accept our apologies.**
 - Code Update:
 - Corrects the button authorship issue
 - Multi-edit of custom-buttons updated to prevent multiple edits of the same button.
 - Minor change to "**Copy Button Name**"
 - Minor change to "**Button Help**"
 - Major update to "**View Custom Button Codes**". Thank you **LouCypher** for this one.