

**Likno Software  
AllWebMenus**

**Server-Side Menus API Add-in**

**Help Document**

**Version 2.0.12 rev 1**

**Check for updated version at:**

**<http://www.likno.com/addins/dhtml-menu-serverside.html>**

<b>HOW TO USE SERVER-SIDE MENUS.....</b>	<b>5</b>
Create Group and Item Styles.....	5
Create the Main Menu .....	6
Add Items.....	7
Add Subgroups .....	8
Produce Menu.....	9
Clarifications on How Server-Side Menus work.....	10
<b>APPENDIX .....</b>	<b>13</b>
ItemStyle Properties.....	13
name.....	13
align0, align1, align2.....	13
textfont0, textfont1, textfont2.....	13
textsize0, textsize1, textsize2.....	14
textdecor0, textdecor1, textdecor2.....	14
color0, color1, color2.....	14
padding0, padding1, padding2.....	15
margin0, margin1, margin2 .....	15
border0, border1, border2 .....	15
borderwidth0, borderwidth1, borderwidth2.....	16
bordercolor0, bordercolor1, bordercolor2.....	16
bgcolor0, bgcolor1, bgcolor2.....	16
bgimage0, bgimage1, bgimage2 .....	17
leftimage0, leftimage1, leftimage2 .....	17
rightimage0, rightimage1, rightimage2.....	18
minwidth .....	18
minheight.....	18
subimage0, subimage1, subimage2 .....	19
subimagewidth0, subimagewidth1, subimagewidth2 .....	19
cursor .....	20
GroupStyle Properties.....	21
name.....	21
type.....	21
itemsWidenHow .....	21
leftMargin.....	22
rightMargin .....	22
groupwidth.....	22
when .....	23
how.....	23
hideHow .....	24
showUnfoldEffect .....	25
hideUnfoldEffect.....	25
showDelay.....	26
hideDelay .....	26
opacity .....	26
bgcolor.....	27
bgimage .....	27
border .....	27
borderwidth.....	28
bordercolor .....	28
itemdistance .....	28
uniform .....	29
alignsWith.....	29

alignoffset .....	29
openDirection .....	30
groupoffset .....	30
scrollbar .....	30
divider .....	31
slidingGroup .....	31
slidingRelativeTo .....	31
slidingLeftMargin .....	32
slidingRightMargin .....	32
slidingTopMargin .....	32
slidingBottomMargin .....	33
slidingIndentation .....	33
slidingShrink .....	33
slidingStretch .....	34
Menu Properties .....	35
visible .....	35
style .....	35
text .....	35
image .....	36
imagealign .....	36
tooltip .....	36
headerstyle .....	37
footertext .....	37
footerimage .....	37
footerimagealign .....	38
footertooltip .....	38
footerstyle .....	38
movable .....	39
drifty .....	39
sticky .....	40
stickyUntilClick .....	40
elementrelative .....	40
corner .....	41
offsetx .....	41
offsety .....	42
scriptappear .....	42
scriptdisappear .....	42
rememberState .....	43
pushPull .....	43
Group Properties .....	44
visible .....	44
style .....	44
text .....	44
image .....	45
imagealign .....	45
tooltip .....	45
headerstyle .....	46
footertext .....	46
footerimage .....	46
footerimagealign .....	47
footertooltip .....	47
footerstyle .....	47
scriptappear .....	48
scriptdisappear .....	48
Item Properties .....	49
visible .....	49
style .....	49
text0, text1, text2 .....	49
image0, image1, image2 .....	50
imagealign0, imagealign1, imagealign2 .....	50

url.....	50
targetframe .....	51
status .....	51
tooltip .....	51
script0, script1, script2.....	52
sepWidth .....	52
sepColor .....	52
sepColorShadow .....	53
sepMargin.....	53
Special Functions.....	54
CopyItemStyle (ItemStyleVariable, NewItemStyleName) .....	54
CopyGroupStyle (GroupStyleVariable, NewGroupStyleName) .....	54
newItemBefore(ItemVariable, ItemRelatedArguments) .....	55

# HOW TO USE SERVER-SIDE MENUS

## Create Group and Item Styles

The first step, in order to create a Server-Side Menu, is to define the styles that the menu, its items and its subgroups will use. There are two types of styles: the "Item Style" that can be assigned to Items and Group header, and the "Group Style" that can be assigned to the Main Menu Group and Subgroups.

Two functions can be used to create styles:

- `ItemStyle()` for item and group headers
- `GroupStyle()` for groups

When creating a new style, it is better to assign the style to a variable, eg:

```
IS1=new ItemStyle();  
or  
GS1=new GroupStyle();
```

This way, you can later assign values to the properties of the style, eg:

```
IS1.bgcolor1="#0055E5";  
or  
GS1.bgcolor="#FFFFFF";
```

Instead of first creating a style, and then assigning values to the style properties, you can assign style properties as you create the style itself. Let's see an example:

```
IS1 = new ItemStyle("name=itemstyle1;color0=#C08040");
```

In this example, we created a new Item Style and assigned it to the `IS1` variable. The value of the property "name" of the style is "itemstyle1" and the value of the property "color0" (the color of normal state) is "#C08040". Another example is:

```
GS1 = new GroupStyle("name=groupstyle1;bgcolor=#A0B1C2");
```

In this example, we created a new Group Style and assigned it to the `GS1` variable. The value of the property "name" of the style is "groupstyle1" and the value of the property "bgcolor" (the background color of the Group) is "#A0B1C2".

This process of assigning values to properties has the following basic rules:

- Use double quotes (") at the beginning and end of the argument.
- Inside the double quotes, define the properties and values in the following form:  
`property1=value1;property2=value2;...;propertyX=valueX`
- Do not use within the argument double quotes. If you wish to do so, define them as `\"` (backslash followed by double quote).

These rules also apply for the creation of the Main Menu, Item and Subgroup.

## Create the Main Menu

The next step is to create the Main Menu. In order to create it we use the following function:

```
Menu ()
```

You should assign the Menu object to a variable, because later you will need to refer to the Menu in order to add Items. An example of a new Menu creation is:

```
mymenu=new Menu ();
```

As with styles, you can assign values to Menu properties as you create the Menu, following the same steps as with the styles. Eg:

```
mymenu=new Menu("style=groupstyle1;text=Menu Text");
```

In this example, the value of the property "style" of the Menu is "groupstyle1" (we previously have created this Group Style) and value of the property "text" is "Menu Text".

The next step is to add Items to a Menu object.

## Add Items

In order to add Items we must use a Menu object or a Group object function called:

```
newItem()
```

Note that we must use the variable that we previously assigned when creating the Menu or Group. For example, to create an item for our Menu we should type:

```
I1=mymenu.newItem();
```

To add one more item we should type:

```
I2=mymenu.newItem();
```

As you notice, we use again variables (I1, I2) to assign the newly created Item objects. This is useful if later we wish to add a Subgroup in an item, or if we wish to change a value of the property of an item.

As with styles and Menu objects, you can assign values to Item properties at the time of the creation, following the same rules as with styles. Eg:

```
I1=mymenu.newItem("style=itemstyle1;text0=Normal  
Text;text1=Mouseover Text");
```

In this example, the value of the property "style" of the Item is "itemstyle1" (we previously have created this Item Style), the value of the property "text0" (text of item in Normal State) is "Normal Text" and the property "text1" (text of item in Mouseover State) is "Mouseover Text".

## Add Subgroups

In order to add Subgroups we must use an Item object function called:

```
newGroup()
```

Note that we must use the variable that we previously created the Item with. For example to create a Subgroup for our created Item we should type:

```
S3=I2.newGroup();
```

We use again a variable (S3) in order to assign the newly created Subgroup object. Later, we could use this variable to add Items, or to change the value of the property of a Subgroup.

As with styles, Menu object and Item object, you can assign values to Subgroup properties as you create your Subgroup, following the same rules as with styles. Eg:

```
S3=I2.newGroup("style=groupstyle1;text=Subgroup Text");
```

In this example, the value of the property "style" of Group is "groupstyle1" (we previously have created this Group Style) and the value of the property "text" (text of item in Normal State) is "Normal Text" and the value of the property "text" is "Subgroup Text".

## Produce Menu

Previously we examined all of the functionality of how to create Item and Group styles, create a Menu, as well as add Items and Subgroups. After the definition of all of the above, we must produce the menu. The function to do so is:

```
ProduceMenu(menuvariable);
```

As you can see the function has one argument only, the "menuvariable". You should pass the variable that you previously created your Menu object. In our examples we use the `mymenu` variable in order to create the Menu object. So, we should type:

```
ProduceMenu(mymenu);
```

Note, that after the call of the `ProduceMenu` function, you can NOT change any property of either the Style, Menu, Item or Group object.

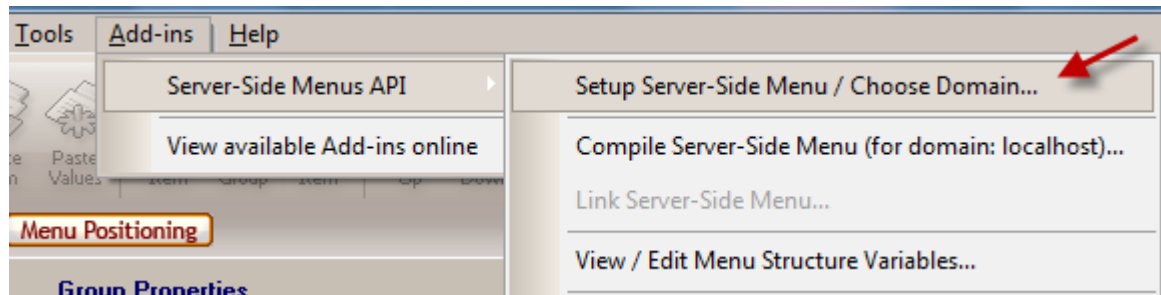
You can also view (and download) a full example online at:

<http://www.likno.com/addins/dhtml-menu-serverside-example.html>

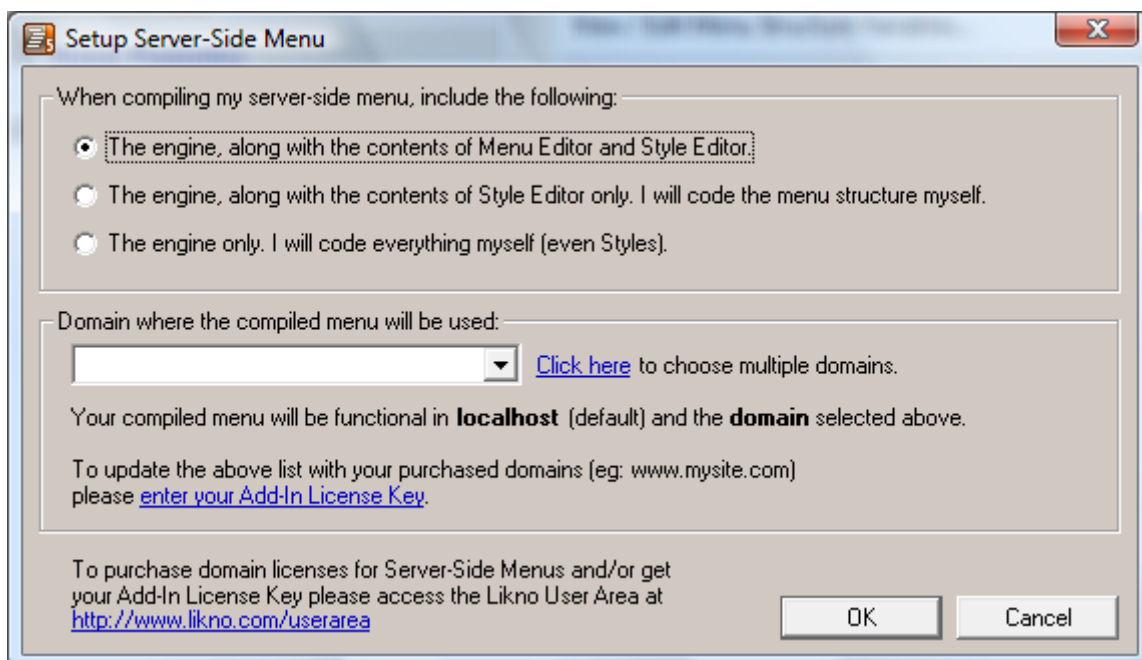
## Clarifications on How Server-Side Menus work

This chapter explains more about how the Server-Side Menus work.

Initially, as usual, you design your Menu by using the AllWebMenus interface. The Menu you create is a Menu that will cover most of the cases in your Server-Side pages. After you finish with the Menu design, it's time to setup your menu. To do this click "Add-ins ->Server-Side Menus->Setup Server-Side Menu".



A popup screen appears, where you can configure your Server-Side Menu.



First, select one of the three options of what the compiled menu will include:

- The engine, along with the contents of Menu Editor and Style Editor.  
This option will create all the commands you need in order to display your menu as a Server-Side Menu. The commands include both Style commands and Menu structure commands.
- The engine, along with the contents of Style Editor only. I will code the menu structure myself.

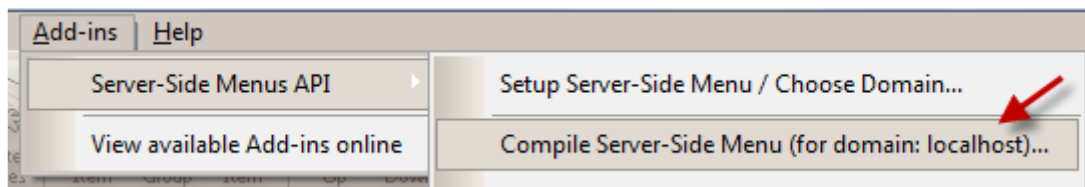
This option will create only the initial commands for your menu, and also the commands for the style of your menu. You should later add all the appropriate commands to build your menu structure.

- The engine only. I will code everything myself (even Styles).  
This option will create only the initial command for your menu. You should later add the appropriate commands for Styles and also the Menu structure.

After that you should choose a Domain name to indicate where the Server-Side Menu will be used. The Domain name appears only after you register.

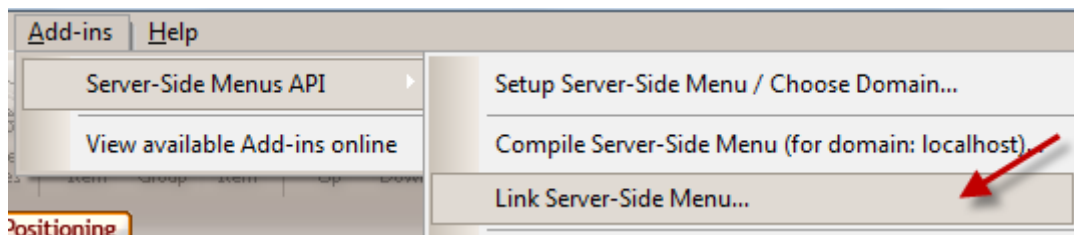
After the Setup of your Server-Side Menu you should compile your menu. Usually, you compile the static menu from "Tools->Compile Menu" or by pressing "Ctrl+F12".

Now that you wish to compile a Server-Side menu, you should click on "Add-ins ->Server-Side Menus->Compile Server-Side Menu".



When you compile the menu, a menu.js file (or whatever filename you choose) will be created. This menu.js file will have all the appropriate commands that create the menu.

After the compilation finishes, you should link the menu to your pages. Beware, that you should link your Server-Side Menu by clicking "Add-ins ->Server-Side Menus->Link Server-Side Menu".



When you link the menu.js file, the program will automatically add the following lines, if they are missing:

```
<script type="text/javascript">
awmBuildMenu();
//---- Add your Server-Side code right after this line. ----
if (typeof(menu)!="undefined") ProduceMenu(menu);
</script>
```

As mentioned before, the `ProduceMenu()` function is the function that displays the menu.

Now it's your turn. If you wish to change any value of any property of an Item, Group, Style or even if you wish to add more Items and Groups, you should type your code just before the call of the `ProduceMenu()` function.

This way you will have:

- A constant menu in the `menu.js` file
- Changes you wish to apply to menu in your Server-Side files (`.asp`, `.php`, `.aspx`, `.jsp`, etc.)

It is important **NOT** to edit the `menu.js` file, because any editing you make, will be lost the next time you recompile your menu.

# APPENDIX

## ItemStyle Properties

### name

**Description** This declares the name of the item style. This is the most important property, since it will be used later when you define an item, in order to declare the style of this item by declaring the value of name property.

**Type** String

**Values** Any String value

**Example** `IS1.name="myitemstylename";`

### align0, align1, align2

**Description** Specifies how the Content (both Text and Image) is aligned within the Item.

**Type** String

**Values** "left" | "center" | "right"

**Default** "center"

**Example** `IS1.align1="right";`

### textfont0, textfont1, textfont2

**Description** Specifies the font of the Item's text.

**Type** String

**Values** Any String that represents a valid font name.

**Default** "sans-serif"

**Example** `IS1.textfont0="Arial";  
IS1.textfont1="Verdana, Arial";`

### textsize0, textsize1, textsize2

**Description** Specifies the font size of the Item's text.

**Type** String

**Values** Any String that represents a valid font size value.

**Default** "14px"

**Example**

```
IS1.textsize0="11pt";
IS1.textsize1="12px";
```

### textdecor0, textdecor1, textdecor2

**Description** Specifies whether the Item's text is shown in Bold, Italic or Underlined.

**Type** String

**Values** null or any combination of "B", "I", "U".

**Default** null

**Example**

```
IS1.textdecor0="B";
IS1.textdecor1="IU";
IS1.textdecor2="BIU";
```

### color0, color1, color2

**Description** Sets the color of the Item's text.

**Type** String

**Values** Any String that begins with '#' followed by any valid html color hexadecimal color.

**Default** "#000000" for color0 and "#FFFFFF" for color1, color2

**Example**

```
IS1.textcolor0="#808080";
IS1.textcolor1="#C0C0C0";
IS1.textcolor2="#4A4A3B";
```

### padding0, padding1, padding2

**Description** Specifies a number of pixels to be added between the Item's content (text and/or image) and the Item's edges.

**Type** Number – Integer

**Values** Any integer number greater or equal than zero.

**Default** 0

**Example** `IS1.padding0=2;`  
`IS1.padding1=4;`

### margin0, margin1, margin2

**Description** Specifies a number of spaces to be added just before and after the Item's text. This creates an indirect "horizontal padding" between the text and the Item's vertical edges or image.

**Type** String

**Values** Any String of the following form:  
"x,y"  
where x and y are integers greater or equal than zero.

**Default** "0,0"

**Example** `IS1.margin0="1,1";`  
`IS1.margin1="2,4";`

### border0, border1, border2

**Description** Specifies the style of the Item's border.

**Type** String

**Values** "none" | "solid" | "outset" | "inset"

**Default** "none"

**Example** `IS1.border0="solid";`  
`IS1.border1="inset";`

### borderwidth0, borderwidth1, borderwidth2

**Description** Specifies the width of the Item's border if the border has been set to a value other than "none".

**Type** Number – Integer

**Values** Any integer greater than or equal to zero.

**Default** 0

**Example** `IS1.borderwidth0=2;`  
`IS1.borderwidth1=1;`

### bordercolor0, bordercolor1, bordercolor2

**Description** Specifies the color of the Item's border if the border has been set to a value other than "none".

**Type** String

**Values** Any String that begins with '#' followed by any valid html color hexadecimal color.

**Default** "#C0C0C0"

**Example** `IS1.bordercolor0="#404040";`  
`IS1.bordercolor1="#C0D0A1";`

### bgcolor0, bgcolor1, bgcolor2

**Description** Specifies the Item's background with a color. If the "null" value has been used, then the background of the Item's Group is shown instead.

**Type** String

**Values** null | any String that begins with '#' followed by any valid html color hexadecimal color.

**Default** "#C0C0C0"

**Example** `IS1.bgcolor0=null;`  
`IS1.bgcolor1="#C1D2E1";`

### bgimage0, bgimage1, bgimage2

**Description** Shows a TILING image on the Item's background. If the "null" value has been used, then the bgcolorX is shown instead.

**Type** String

**Values** null | any String that declares the image name stored in "Menu images" folder.

**Default** Null

**Example** IS1.bgimage0="hearts.jpg";  
IS1.bgimage1="arrow.gif";  
IS1.bgimage2=null;

### leftimage0, leftimage1, leftimage2

**Description** Shows a FIXED image on the Item's background, aligned to the left.

**Type** String

**Values** Null | any String that declares the image name stored in "Menu images" folder.

**Default** Null

**Example** IS1.leftimage0="hearts.jpg";  
IS1.leftimage1="arrow.gif";  
IS1.leftimage2=null;

### rightimage0, rightimage1, rightimage2

**Description** Shows a FIXED image on the Item's background, aligned to the right.

**Type** String

**Values** Null | any String that declares the image name stored in "Menu images" folder.

**Default** Null

**Example**

```
IS1.rightimage0="hearts.jpg";  
IS1.rightimage1="arrow.gif";  
IS1.rightimage2=null;
```

### minwidth

**Description** Sets the minimum width for the Item's size, excluding border size.

**Type** Number – Integer

**Values** Any integer greater than or equal to zero.

**Default** 0

**Example**

```
IS1.minwidth=220;
```

### minheight

**Description** Sets the minimum height for the Item's size, excluding border size.

**Type** Number – Integer

**Values** Any integer greater than or equal to zero.

**Default** 0

**Example**

```
IS1.minheight=93;
```

### subimage0, subimage1, subimage2

**Description** Adds an image to emphasize the existence of a submenu linked to the item. The image appears only if there is a subgroup linked to the item.

Also, if you set the `subimageX` property, don't forget to set the appropriate `subimagewidthX` property. If you don't set it the `subimageX` will not appear.

**Type** String

**Values** Null | any String that declares the image name stored in "Menu images" folder.

**Default** Null

**Example**

```
IS1.subimage0="hearts.jpg";
IS1.subimage1="arrow.gif";
IS1.subimage2=null;
```

### subimagewidth0, subimagewidth1, subimagewidth2

**Description** Specifies the width of the relative `subimageX` that was specified. If you do not set the relative `subimageX`, this value will not be used anywhere.

**Type** Number – Integer

**Values** Any integer greater than or equal to zero.

**Default** 0

**Example**

```
IS1.subimagewidth0=14;
IS1.subimagewidth1=21;
```

## cursor

**Description** Specifies the cursor to be shown when the mouse passes over the Item.

**Type** String

**Values** "default" | "hand" | "crosshair" | "help" | "move" | "text" | "wait"

**Default** "default"

**Example** `IS1.cursor="move";`

## GroupStyle Properties

### name

**Description** This declares the name of the group style. This is the most important property, since it will be used later when you define a group, in order to declare the style of this group by declaring the value of name property.

**Type** String

**Values** Any String value

**Example** `GS1.name="mygroupstylename";`

### type

**Description** Specifies whether the Group is Vertical or Horizontal.

**Type** Number – Integer

**Values** 0 | 1 | 2  
0: Vertical Group,  
1: Horizontal Group,  
2: Horizontal Widen Group to fit entire window width

**Default** 0

**Example** `GS1.type=1;`

### itemsWidenHow

**Description** Specifies the way that items are aligned inside a Horizontal Widen Group. Note that this value is effective only if the type property has a value of 2

**Type** Number

**Values** 0: Do not widen items, show background on the right  
1: Do not widen items, show background around  
2: Do not widen items, show background on the left  
3: Widen group, to fit entire width

**Default** 0

**Example** `GS1.itemsWidenHow=1;`

### leftMargin

**Description** Specifies the left Margin of a Horizontal Widen Group. Note that this value is effective only if the type property has a value of 2

**Type** Number or String

**Values** Any positive integer or Percent

**Default** 0

**Example** `GS1.leftMargin="15%";`

### rightMargin

**Description** Specifies the right Margin of a Horizontal Widen Group. Note that this value is effective only if the type property has a value of 2

**Type** Number or String

**Values** Any positive integer or Percent

**Default** 0

**Example** `GS1.rightMargin=125;`

### groupwidth

**Description** Specifies the exact width of the Group in pixels. Note, that it only works for Vertical Groups.

**Type** Number – Integer

**Values** Any integer greater than or equal to zero.

**Default** 0

**Example** `GS1.groupwidth=120;`

## when

**Description** Specifies when the Group appears. When passing over or when clicking on the Group's parent item.

**Type** Number – Integer

**Values** 0 | 1  
(0 is for when mouse over, 1 for when mouse click)

**Default** 0

**Example** GS1.when=1;

## how

**Description** Specifies how the Group appears.

**Type** Number – Integer

**Values** 0 - 19  
where:  
0: Instantly  
1: Unfolding 1  
2: Unfolding 2  
3: Barn  
4: Blinds  
5: Checker Board  
6: Fade  
7: Iris - Circle  
8: Iris - Cross  
9: Iris - Plus  
10: Iris - Square  
11: Iris - Star  
12: Pixelate  
13: Radial Wipe  
14: Random Bars  
15: Random Dissolve  
16: Slide  
17: Spiral  
18: Strecth  
19: Wheel  
(The values from 3 to 19 default to value 2  
(Unfolding 2) on all browsers except IE5.5+.

**Default** 0

**Example** GS1.how=5;

## hideHow

**Description** Specifies how the Group disappears.

**Type** Number – Integer

**Values** 0 - 19

where:

- 0: Instantly
- 1: Unfolding 1
- 2: Unfolding 2
- 3: Barn
- 4: Blinds
- 5: Checker Board
- 6: Fade
- 7: Iris - Circle
- 8: Iris - Cross
- 9: Iris - Plus
- 10: Iris - Square
- 11: Iris - Star
- 12: Pixelate
- 13: Radial Wipe
- 14: Random Bars
- 15: Random Dissolve
- 16: Slide
- 17: Spiral
- 18: Strecth
- 19: Wheel

(The values from 3 to 19 default to value 2  
(Unfolding 2) on all browsers except IE5.5+.

**Default** 0

**Example** `GS1.hideHow=7;`

### showUnfoldEffect

**Description** Specifies the unfold direction. The value applies only if Unfold 1 or Unfold 2 effect has been chosen in “how”.

**Type** Number – Integer

**Values** 0 - 10  
where:  
0: Auto  
1: Right  
2: Left  
3: Downwards  
4: Upwards  
5: Right-Downwards  
6: Right-Upwards  
7: Left-Downwards  
8: Left-Upwards  
9: Center  
10: Unfold 3

**Default** 0

**Example** `GS1.showUnfoldEffect=3;`

### hideUnfoldEffect

**Description** Specifies the unfold direction. The value applies only if Unfold 1 or Unfold 2 effect has been chosen in “hideHow”.

**Type** Number – Integer

**Values** 0 - 10  
where:  
0: Auto  
1: Right  
2: Left  
3: Downwards  
4: Upwards  
5: Right-Downwards  
6: Right-Upwards  
7: Left-Downwards  
8: Left-Upwards  
9: Center  
10: Unfold 3

**Default** 0

**Example** `GS1.hideUnfoldEffect=3;`

### showDelay

**Description** Specifies after how many milliseconds the group appears.

**Type** Number – Integer

**Values** Any integer greater than or equal to zero.

**Default** 200

**Example** `GS1.showDelay=750;`

### hideDelay

**Description** Specifies after how many milliseconds the group disappears.

**Type** Number – Integer

**Values** Any integer greater than or equal to zero.

**Default** 200

**Example** `GS1.hideDelay=250;`

### opacity

**Description** A value of 0 opacity makes the Group fully transparent.  
A value of 100 opacity makes the Group fully opaque.

**Type** Number – Integer

**Values** 0 – 100

**Default** 100

**Example** `GS1.opacity=80;`

### bgcolor

**Description** Fills the Group's background with a color or shows it transparent if the "null" value is used.

**Type** String

**Values** null | any String that begins with '#' followed by any valid html color hexadecimal color.

**Default** "#FFFFFF"

**Example** GS1.bgcolor="#808080";

### bgimage

**Description** Shows a TILING image on the Group's background. If "null" value used, then the bgcolor is shown instead.

**Type** String

**Values** null | any String that declares the image name stored in "Menu images" folder.

**Default** null

**Example** GS1.bgimage="hearts.jpg";

### border

**Description** Specifies the style of the Group's border.

**Type** String

**Values** "none" | "solid" | "outset" | "inset"

**Default** "solid"

**Example** GS1.border="inset";

### borderwidth

**Description** Specifies the width of the Group's border if border has been set to a value other than "none".

**Type** Number – Integer

**Values** Any integer greater than or equal to zero.

**Default** 1

**Example** `GS1.borderwidth=2;`

### bordercolor

**Description** Specifies the color of the Group's border if border has been set to a value other than "none".

**Type** String

**Values** Any String that begins with '#' followed by any valid html color hexadecimal color.

**Default** "#000000"

**Example** `GS1.bordercolor="#404040";`

### itemdistance

**Description** Specifies the distance between the adjacent items of the Group.

**Type** Number – Integer

**Values** Any integer greater than or equal to zero.

**Default** 0

**Example** `GS1.itemdistance=2;`

### uniform

**Description** All items have the same size as determined by the biggest item.

**Type** Number – Integer

**Values** 0 | 1  
(0 means "no", 1 means "yes")

**Default** 0

**Example** `GS1.uniform=1;`

### alignsWith

**Description** Specifies how the Group aligns with its parent item or group.

**Type** Number – Integer

**Values** 0 - 3  
where:  
0: Parent Item  
1: Top/Left edge of parent group  
2: Bottom/Right edge of parent group  
3: Centred to parent group

**Default** 0

**Example** `GS1.alignsWith=3;`

### alignoffset

**Description** Specifies the alignment offset of the Group with its parent item.

**Type** Number – Integer

**Values** Any integer.

**Default** 0

**Example** `GS1.alignoffset=-5;`

### openDirection

**Description** Specifies where the Group appear relatively to its parent Group.

**Type** Number – Integer

**Values** 0 - 2  
where:  
0: Auto  
1: Bottom/Right edge of parent group  
2: Top/Left edge of parent group

**Default** 0

**Example** `GS1.openDirection=2;`

### groupoffset

**Description** Specifies the distance of the Group from its parent item.

**Type** Number – Integer

**Values** Any integer.

**Default** 0

**Example** `GS1.groupoffset=-2;`

### scrollbar

**Description** Specifies whether a group has vertical scrollbar or not.  
**NOTE:** It applies only to vertical groups.

**Type** Number – Integer

**Values** -1: Scrollbar appears when group height exceeds browser window height.  
0: means no scrollbar.  
>0: A scrollbar will appear at the specified height.

**Default** -1

**Example** `GS1.scrollbar=50;`

### divider

**Description** Specifies the number of rows (for horizontal) or columns (for vertical) a group has.

**Type** Number – Integer

**Values** Any integer greater than or equal to 1.

**Default** 1

**Example** `GS1.divider=2;`

### slidingGroup

**Description** Specifies whether a Group is Sliding or not.

**Type** Number – Integer

**Values** 0, 1  
where:  
0: is not Sliding  
1: is Sliding

**Default** 0

**Example** `GS1.slidingGroup=1;`

### slidingRelativeTo

**Description** Specifies how the Group appears relatively to its parent Item (the item that opened it).

**Type** Number – Integer

**Values** 0 - 2  
where:  
0: Left Bottom Corner of parent Item  
1: Center Bottom Corner of parent Item  
2: Right Bottom Corner of parent Item

**Default** 0

**Example** `GS1.slidingRelativeTo=2;`

### slidingLeftMargin

**Description** Specifies the left Margin of a Sliding Group.

**Type** Number

**Values** Any positive integer

**Default** 0

**Example** `GS1.slidingLeftMargin=9;`

### slidingRightMargin

**Description** Specifies the right Margin of a Sliding Group.

**Type** Number

**Values** Any positive integer

**Default** 0

**Example** `GS1.slidingRightMargin=9;`

### slidingTopMargin

**Description** Specifies the top Margin of a Sliding Group.

**Type** Number

**Values** Any positive integer

**Default** 0

**Example** `GS1.slidingTopMargin=5;`

### slidingBottomMargin

**Description** Specifies the bottom Margin of a Sliding Group.

**Type** Number

**Values** Any positive integer

**Default** 0

**Example** `GS1.slidingBottomMargin=3;`

### slidingIndentation

**Description** Specifies the Indentation of a Sliding Group.

**Type** Number

**Values** Any positive integer

**Default** 10

**Example** `GS1.slidingIndentation=15;`

### slidingShrink

**Description** Specifies whether a Group will shrink to fit its parent Group, if this Group is wider.

**Type** Number – Integer

**Values** 0, 1  
where:  
0: Group will not shrink  
1: Group will shrink

**Default** 0

**Example** `GS1.slidingShrink=1;`

## slidingStretch

**Description** Specifies whether a Group will stretch to fit its parent Group, if this Group is narrower.

**Type** Number – Integer

**Values** 0, 1  
where:  
0: Group will not shrink  
1: Group will shrink

**Default** 0

**Example** `GS1.slidingStretch=1;`

## Menu Properties

### visible

**Description** Specifies whether the menu will appear in a page or not. If you specify it to 0, then no menu will appear at all.

**Type** Number – Integer

**Values** 0 | 1  
(0 means "no", 1 means "yes")

**Default** 1

**Example** `mymenu.visible=0;`

### style

**Description** Specifies the Group Style name that the Menu Group will use. If not specified then the first Group Style will be used.

**Type** String

**Values** Any String value that reflects to a Group Style that has been previously created.

**Example** `mymenu.style="mygroupstylename";`

### text

**Description** Specifies the text shown on the Header of the Group. If it is not specified, then no Header will be used at all, unless an "image" is declared. Also, if you wish to remove the text, then assign the `null` value and not an empty string ("").

**Type** String

**Values** Any String value

**Default** Null

**Example** `mymenu.text="Main Menu Header";`

### image

**Description** Specifies the image shown on the Header of the Group.

**Type** String

**Values** null | any String that declares the image name stored in "Menu images" folder.

**Default** Null

**Example** `mymenu.image="headerimg.jpg";`

### imagealign

**Description** Specifies how the image is aligned relatively to text.

**Type** String

**Values** "left" | "right" | "top" | "bottom"

**Default** "right"

**Example** `mymenu.imagealign="top";`

### tooltip

**Description** Specifies the text to be displayed as a tooltip when the user pauses the mouse pointer over the Header, if a Header exists.

**Type** String

**Values** Any String value

**Default** Null

**Example** `mymenu.tooltip="Main Menu Tooltip";`

### headerstyle

**Description** Specifies the Item Style name that the header item will use. If it is not specified then the first Item Style will be used.

**Type** String

**Values** Any String value that reflects to an Item Style that has been previously created.

**Example** `mymenu.headerstyle="myitemstylename";`

### footertext

**Description** Specifies the text shown on the Footer of the Group. If it is not specified, then no Footer will be used at all, unless a "footerimage" is declared. Also, if you wish to remove the footertext, then assign the null value and not an empty string ("").

**Type** String

**Values** Any String value

**Default** Null

**Example** `mymenu.footertext="Main Menu Footer";`

### footerimage

**Description** Specifies the image shown on the Footer of the Group.

**Type** String

**Values** null | any String that declares the image name stored in "Menu images" folder.

**Default** Null

**Example** `mymenu.footerimage="footerimg.jpg";`

### footerimagealign

**Description** Specifies how the footer image is aligned relatively to footer text.

**Type** String

**Values** "left" | "right" | "top" | "bottom"

**Default** "right"

**Example** `mymenu.footerimagealign="top";`

### footertooltip

**Description** Specifies the text to be displayed as a tooltip when the user pauses the mouse pointer over the Footer, if a Footer exists.

**Type** String

**Values** Any String value

**Default** Null

**Example** `mymenu.footertooltip="Menu Footer Tooltip";`

### footerstyle

**Description** Specifies the Item Style name that the footer item will use. If it is not specified then the first Item Style will be used.

**Type** String

**Values** Any String value that reflects to an Item Style that has been previously created.

**Example** `mymenu.footerstyle="myitemstylename";`

## movable

**Description** Specifies whether the user will be able to relocate the menu on the Web Page. This property has an effect only if a header exists, otherwise it is ignored.

**Type** Number – Integer

**Values** 0 | 1  
(0 means "no", 1 means "yes")

**Default** 0

**Example** `mymenu.movable=1;`

## drifty

**Description** Specifies whether the menu can drift on the page when the user scrolls the page.

**Type** Number – Integer

**Values** 0 | 1 | 2 | 3 | 4 | 5 | 6  
where:  
0: No drifty  
1: Instantly on both scrolls  
2: Instantly on vertical scroll only  
3: Instantly on horizontal scroll only  
4: Smoothly on both scrolls  
5: Smoothly on vertical scroll only  
6: Smoothly on horizontal scroll only

**Default** 0

**Example** `mymenu.drifty=4;`

### sticky

**Description** Specifies whether subgroups can keep their open status when the mouse moves away from the menu area.

**Type** Number – Integer

**Values** 0 | 1  
(0 means "no", 1 means "yes")

**Default** 0

**Example** `mymenu.sticky=1;`

### stickyUntilClick

**Description** The value applies only if sticky property is set to 1. Specifies whether the open subgroups disappear on click on the document, or not.

**Type** Number – Integer

**Values** 0 | 1  
(0 means "no", 1 means "yes")

**Default** 1

**Example** `mymenu.stickyUntilClick=1;`

### elementrelative

**Description** Specifies if the menu will be positioned relatively to an element.

**Type** Number – Integer

**Values** 0 | 1  
(0 means "no", 1 means "yes")

**Default** 0

**Example** `mymenu.elementrelative=1;`

## corner

**Description** Specifies the corner that the menu will be positioned relative to Window or Element, depending on the value of an `element relative` property.

**Type** Number – Integer

**Values** 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8

where:

0: Top-left  
1: Top-right  
2: Bottom-right  
3: Bottom-left  
4: Center-left  
5: Top-center  
6: Center-right  
7: Bottom-center  
8: Center-center

**Default** 0

**Example** `mymenu.corner=8;`

## offsetx

**Description** Specifies the horizontal offset for the positioning of the menu.

**Type** Number – Integer

**Values** Any integer.

**Default** 0

**Example** `mymenu.offsetx=-7;`

### offsety

**Description** Specifies the vertical offset for the positioning of the menu.

**Type** Number – Integer

**Values** Any integer.

**Default** 0

**Example** `mymenu.offsety=-3;`

### scriptappear

**Description** Specifies a javascript code to be executed when the Menu appears.

**Type** String

**Values** Any String that represents valid JavaScript.

**Default** Null

**Example** `mymenu.scriptappear="alert('Message: Menu appears')";`

### scriptdisappear

**Description** Specifies a javascript code to be executed when the Menu disappears.

**Type** String

**Values** Any String that represents valid JavaScript.

**Default** Null

**Example** `mymenu.scriptdisappear="alert('Message: Menu disappears')";`

## rememberState

**Description** If you are using Sliding Menus then this option defines whether the menu remembers the menu state (expanded/collapsed) when browsing through different pages.

**Type** Number – Integer

**Values** 0, 1  
where:  
0: option is disabled  
1: option is enabled

**Default** 0

**Example** `GS1.rememberState=1;`

## pushPull

**Description** If you are using Sliding Menus then this option defines whether the menu will push (or pull) page content when sliding menu appear (or disappear).

**Type** Number – Integer

**Values** 0, 1  
where:  
0: option is disabled  
1: option is enabled

**Default** 0

**Example** `GS1.pushPull=1;`

## Group Properties

### visible

**Description** Specifies whether the Group will appear or not. If you specify it to 0, then the Group and ultimately all Items will not appear at all.

**Type** Number – Integer

**Values** 0 | 1  
(0 means "no", 1 means "yes")

**Default** 1

**Example** `S1.visible=0;`

### style

**Description** Specifies the Group Style name that the Group will use. If it is not specified then the first Group Style will be used.

**Type** String

**Values** Any String value that reflects to a Group Style that has been previously created.

**Example** `S1.style="mygroupstylename";`

### text

**Description** Specifies the text shown on the Header of the Group. If it is not specified, then no Header will be used at all, unless an "image" is declared. Also, if you wish to remove the text, then assign the `null` value and not an empty string ("").

**Type** String

**Values** Any String value

**Default** `Null`

**Example** `S1.text="Group Header";`

### image

**Description** Specifies the image shown on the Header of the Group.

**Type** String

**Values** null | any String that declares the image name stored in "Menu images" folder.

**Default** Null

**Example** `S1.image="groupheader.jpg";`

### imagealign

**Description** Specifies how the image is aligned relatively to the text.

**Type** String

**Values** "left" | "right" | "top" | "bottom"

**Default** "right"

**Example** `S1.imagealign="bottom";`

### tooltip

**Description** Specifies the text to be displayed as a tooltip when the user pauses the mouse pointer over the Header of the Group, if a Header exists.

**Type** String

**Values** Any String value

**Default** Null

**Example** `S1.tooltip="Group Header Tooltip";`

### headerstyle

**Description** Specifies the Item Style name that the Header will use. If it is not specified then the first Item Style will be used.

**Type** String

**Values** Any String value that reflects to an Item Style that has been previously created.

**Example** `S1.headerstyle="myitemstylename";`

### footertext

**Description** Specifies the text shown on the Footer of the Group. If it is not specified, then no Footer will be used at all, unless a "footerimage" is declared. Also, if you wish to remove the footer text, then assign the null value and not an empty string ("").

**Type** String

**Values** Any String value

**Default** Null

**Example** `S1.footertext="Group Footer";`

### footerimage

**Description** Specifies the image shown on the Footer of the Group.

**Type** String

**Values** null | any String that declares the image name stored in "Menu images" folder.

**Default** Null

**Example** `S1.footerimage="groupfooter.jpg";`

### footerimagealign

**Description** Specifies how the footer image is aligned relatively to the text.

**Type** String

**Values** "left" | "right" | "top" | "bottom"

**Default** "right"

**Example** `S1.footerimagealign="bottom";`

### footertooltip

**Description** Specifies the text to be displayed as a tooltip when the user pauses the mouse pointer over the Footer of the Group, if a Footer exists.

**Type** String

**Values** Any String value

**Default** Null

**Example** `S1.footertooltip="Group Footer Tooltip";`

### footerstyle

**Description** Specifies the Item Style name that the Footer will use. If it is not specified then the first Item Style will be used.

**Type** String

**Values** Any String value that reflects to an Item Style that has been previously created.

**Example** `S1.footerstyle="myitemstylename";`

### scriptappear

**Description** Specifies a javascript code to be executed when the Group appears.

**Type** String

**Values** Any String that represents valid JavaScript.

**Default** Null

**Example** `S1.scriptappear="alert('Group appears')";`

### scriptdisappear

**Description** Specifies a javascript code to be executed when the Group disappears.

**Type** String

**Values** Any String that represents valid JavaScript.

**Default** Null

**Example** `S1.scriptdisappear="alert('Group disappears')";`

## Item Properties

### visible

**Description** Specifies whether the Item will appear or not. If you specify it to 0, then the Item and any existing subgroup(s) will not appear at all.

**Type** Number – Integer

**Values** 0 | 1  
(0 means "no", 1 means "yes")

**Default** 1

**Example** `I2.visible=0;`

### style

**Description** Specifies the Item Style name that the Item will use. If it is not specified then the first Item Style will be used.

**Type** String

**Values** Any String value that reflects to an Item Style that has been previously created.

**Example** `I2.style="myitemstylename";`

### text0, text1, text2

**Description** Specifies the text shown on the Item.

**Type** String

**Values** (null | Any String value)

**Default** Null

**Example** `I2.text0="Normal text";`  
`I2.text1="Mouse over text";`

### image0, image1, image2

**Description** Specifies the image shown on the Item.

**Type** String

**Values** (null | any String that declares the image name stored in "Menu images" folder)

**Default** Null

**Example**

```
I2.image0="imagenormal.jpg";  
I2.image1="imageover.gif";
```

### imagealign0, imagealign1, imagealign2

**Description** Specifies how the image is aligned relatively to text.

**Type** String

**Values** "left" | "right" | "top" | "bottom"

**Default** "right"

**Example**

```
I2.imagealign0="bottom";  
I2.imagealign1="left";
```

### url

**Description** Specifies the URL (absolute or relative) that opens when clicking on the Item.

**Type** String

**Values** (null | any String value.

**Default** Null

**Example**

```
I2.url="http://www.likno.com";
```

### targetframe

**Description** Specifies the frame or window where the url opens.

**Type** String

**Values** (null | any String value)

**Default** null

**Example** `I2.targetframe="_new";`

### status

**Description** Specifies the text appearing on the Status Bar when the mouse passes over the item.

**Type** String

**Values** (null | any String value)

**Default** Null

**Example** `I2.status="Item Status Bar Text";`

### tooltip

**Description** Specifies the text to be displayed as a tooltip when the user pauses the mouse pointer over the Item.

**Type** String

**Values** (null | any String value)

**Default** null

**Example** `I2.tooltip="Item Tooltip Text";`

### script0, script1, script2

**Description** Specifies the JavaScript code to be executed when the mouse enters, leaves or clicks the Item.

**Type** String

**Values** (null | Any String that represents valid JavaScript.)

**Default** Null

**Example** I2.script1="Mouse over text";  
I2.script2="Mouse click text";

### sepWidth

**Description** Specifies the width of an item's separator.

**Type** String

**Values** (null | Integer value followed by Percent (%) String.)

**Default** Null

**Example** I2.sepWidth="80%";

### sepColor

**Description** Sets the primary color of the Item's separator.

**Type** String

**Values** Any String that begins with '#' followed by any valid html color hexadecimal color.

**Default** Null (no color)

**Example** IS1.sepColor="C0CAC0";

### sepColorShadow

**Description** Sets the shadow color of the Item's separator.

**Type** String

**Values** Any String that begins with '#' followed by any valid html color hexadecimal color.

**Default** Null (no color)

**Example** `I1.sepColorShadow="FFC3A0";`

### sepMargin

**Description** Specifies the margin of an item's separator.

**Type** Number – Integer

**Values** Any positive integer.

**Default** 0

**Example** `I2. sepMargin=3;`

## Special Functions

### CopyItemStyle (ItemStyleVariable, NewItemStyleName)

You can use this function, if you wish to create a new ItemStyle object based on an existing ItemStyle object. The function has two arguments:

ItemStyleVariable	The type of this argument is ItemStyle object. This is the variable from which all item style properties will be copied from.
NewItemStyleName	The type of this argument is string. This argument assigns its value to the <code>name</code> property of the new ItemStyle object.

For example:

```
IS2=CopyItemStyle(IS1, "myitemstyle2");
```

In this example we create a new ItemStyle object which is an exact copy of the `IS1` ItemStyle object. We assign this new ItemStyle object to the `IS2` variable. Also the value of the `name` property of the new ItemStyle is `myitemstyle2`.

### CopyGroupStyle (GroupStyleVariable, NewGroupStyleName)

You can use this function, if you wish to create a new GroupStyle object based on an existing GroupStyle object. The function has two arguments:

GroupStyleVariable	The type of this argument is GroupStyle object. This is the variable from which all group style properties will be copied from.
NewGroupStyleName	The type of this argument is string. This argument assigns its value to the <code>name</code> property of the new GroupStyle object.

For example:

```
GS2=CopyGroupStyle(GS1, "mygroupstyle2");
```

In this example we create a new GroupStyle object which is an exact copy of the `GS1` GroupStyle object. We assign this new GroupStyle object to the `GS2` variable. Also the value of the `name` property of the new GroupStyle is `mygroupstyle2`.

### newItemBefore(ItemVariable, ItemRelatedArguments)

You can use this function, in order to add an item before an already declared item. This is a function of Menu and Submenu object. The function has two arguments:

ItemVariable	The type of this argument is Item object. This is the item variable in which the new item will be created before.
ItemRelatedArguments	The type of this argument is string. This argument is used in the same way as with the <code>newItem</code> function.

In the following example:

```
I8=menu.newItemBefore(I2,"style=itemstyle1,text0=Item8");
```

we create an item, before I2 item in the menu object.

Also, in the following example:

```
I8=S4.newItemBefore(I5,"style=itemstyle1,text0=Item8");
```

we create an item before I5 item in the S4 Submenu object.