Documentation

Introduction

The best way to learn how to use something is to get your hands dirty and do it. Therefore this documentation isn't very long, but instead provides merely basic instructions. Then move on to the Command Reference where you want dig deep. That way you'll learn to do all the cool stuff yourself.

2. Preparing the page

First of all you need to prepare the page you will be using overLIB on. Add the following line in the HEAD section just somewhere prior to the closing tag, </head>:

```
<script type="text/javascript" src="overlib.js"><!-- overLIB (c) Erik Bosrup --></script>
```

Then place the following line inside the <body></body> section of your page--preferably immediately after the body section starts, i.e., just after the <body> tag (**this is important!**):

```
<div id="overDiv" style="position:absolute; visibility:hidden; z-index:1000;"></div>
```

If you have other div layers (i.e., positioned DIV containers) on the page you might want to change the z-index value to avoid overlaps. You must also change overlib.js to the correct path if the files are not placed in the same directory.

3. Two different behaviours

There are two different types of popups that you can make with overLIB. The first type is when the popup goes away after you move your mouse away from a link or other object. The other type 'sticks' around afterwards. We call these "sticky" links. See the difference here and here.

4. Tooltip popups

Let's look into the code for the popup that went away quickly, or a tooltip which it often is called.

```
<a href="javascript:void(0);" onmouseover="return overlib('This is an ordinary
popup.');" onmouseout="return nd();">here</a>
```

As you see this is no ordinary hyperlink as we've placed javascript:void(0); instead of an url as href (one can also use javascript:; if they wish). This causes the browser not to load a new page when we click on the link. This is of course optional. You can link to a page or anything else if you want.

In the onmouseover part overLIB comes in. Whichever type of popup you want to make you call the overlib() function. It can take any amount of parameters. You'll understand why that is good soon. To make a simple popup we just pass the text we want to display to the function inside SINGLE quotes. If the string contains any embedded single quotes they must be escaped by prefacing them with the backslash (\'). Any double quotes (") in the string should be replaced by the entity reference, ".

After that comes onmouseout. It contains one cryptic function that has to be there. The call to the nd() function makes the popup go away when you move the mouse outside the link. You still have to have it there when making stickies though.

5. Sticky popups

Now let's look at the popup that stayed after the mouse was moved away from the link - the sticky popup.

```
<a href="javascript:void(0);" onmouseover="return overlib('This is what
we call a sticky, since I stick around (it goes away if you move the mouse
OVER and then OFF the overLIB popup--or mouseover another overLIB).',
STICKY, MOUSEOFF);" onmouseout="return nd();">here</a>
```

The only difference between the two of them (except for the text passed) is that the 'sticky' one has more text after the text inside quotes, which are called **commands**. In this case it is the STICKY command which tells overLIB that we want the popup to stick around. And the MOUSEOFF command which allows you to close the popup by first mousing onto it and then off of it. NOTE: all **commands** are in UPPERCASE and can contain zero or more parameter values of different types. See the Command Reference page for a description of the core module commands.

6. Clickable popups and commands

Now that you know the really basic parts lets look at a bit tougher stuff. You remember from the frontpage that you could click on links and make a popup show up then? This isn't hard either. You just place the overLIB code in onclick instead of onmouseover. (Or both.) Here is an example: Click here! Lets digest the code again:

```
<a href="javascript:void(0);" onclick="return overlib('This is a sticky
 with a caption. And it is centered under the mouse!', STICKY, CAPTION,
 'Sticky!', CENTER);" onmouseout="nd();">Click here!</a>
```

Here we have the overlib() call in onclick instead of onmouseover, and we have lots more commands! Just as STICKY is a command that tells overLIB to make the popup 'sticky', the other commands tell overLIB to do other things. Normally, the first command to a popup is a text string inside of single quotes, but it is not always true. (See the detailed discussion contained in the Command Reference page). Following text, you can pass any command you want to overLIB, in any order you want. Some commands expect to receive more than just a command though, it also wants some data (like CAPTION, 'Insert witty caption here'. When that is the case the next parameter is expected to be that data. Some commands also expect more than just one parameter. (These are explained in the Command Reference).

Lets go through the commands in this call to see what actually happens. First we have STICKY that tells overLIB that this popup should be a sticky and stay around. Followed by that is CAPTION and its parameter 'Sticky!'. This tells overLIB that we want a CAPTION on the popup and that the caption text should be **Sticky!**. After that the last command is CENTER which tells overLIB to center the popup under the mouse pointer.

7. Customizing overLIB

You can customize overLIB in three different ways. The first method you already know: by passing commands. However if you want all the popups to be red and use a larger font, it isn't very fun to pass a bunch of commands for each and every popup. The commands you pass last for that specific popup. Commands are intended when you want something different than the ordinary.

- **In overlib.js**
  To set site-wide defaults (used by all pages using the same overlib.js) open up overlib.js in your favourite text editor. In it, look for "Configuration". There you can change overLIB's JavaScript default variable settings. The variables you should change all start with ol_ and there is a description on how you change each of them. They look like this:

  ```
  // Font face for the main text
  if (typeof ol_textfont == 'undefined')
  { var ol_textfont="Verdana,Arial,Helvetica";}
  ```

  In the Command Reference you can also see which variable each command overrides. Changing settings in the "Configuration" section of overlib.js is an effective way of modifying overLIB to suit your needs.

- **On a page**
  When you need to make changes to many popups on one specific page, you can do that on the page without affecting other pages in your site. Simply assign the variable you want to have a page default for and place that JavaScript code before loading overlib.js. Here's how it looks:

  ```
  <script type="text/javascript">
  var ol_textfont = "Courier New, Courier";
  </script>
  <div id="overDiv" style="position:absolute; visibility:hidden;
  z-index:1000;"></div>
  <script language="JavaScript" src="overlib.js"><!-- overLIB
  (c) Erik Bosrup --></script>
  ```

- When calling overlib()
  Finally (as previously mentioned), you can pass commands to in the overlib() call itself to change settings as well. Here is the previous example as a command instead:

  ```
  <a href="javascript:void(0);" onmouseover="return
  overlib('Different font.',TEXTFONT,'Courier, Courier New');"
  onmouseout="return nd();">My link</a>
  ```

Each of these methods, along with use of the routine, `overlib_pagedefaults`, is explained further in the Command Reference.

8. Command reference

Now that you know the basics how the overlib() function works, as well as how you can change the settings in the script or on the page as well, you are ready to start having fun with all the advanced features overLIB has to offer. The Command Reference has all the commands overLIB understands (there are more than 50!) and it is where you will find information on how to use them.

9. overLIB Plugins

overLIB also has the capacity to be enhanced, through the use of Plugins. Plugins provide access to many more commands that a user may want his/her popups to have. The documentation for each plugin explains the commands each particular plugin supports. If you try to use a command without having its supporting module loaded, you'll get an error saying that XXXX is undefined. Also, if you find that overLIB doesn't have the specific, mission critical feature you're mossing, you can always write a plugin module yourself!

10. Still don't understand?

Don't worry, you're not the first. Before you start looking for help though, try going through the manual again and trying out the examples etc. Also make sure you use an "empty" page that doesn't have any other JavaScript's, applets and the like on it. Make a page with just overLIB first and then try the more complex things.

When things just don't work, there are ways of getting help. Take a look at overLIB's Support page and see what fits you.

Welcome to overLIB!
