June 2004 - Impure Sorcery

Merry meet, fellow RoboNetizens!

What is this sorcery you say? Why, why, RoboWizard! You have tricked us! In your Who is this RoboWizard? topic, you told us straight out that you weren't into such things!

Relax, take a chill pill, and allow the RoboWizard to elaborate on what is meant by "Impure Sorcery". In this context, I am referring to a set of circumstances that seems to occur with alarming frequency. This would be the most unfortunate circumstance of finding one's self in possession of WebHelp output files as the only avenue to a RoboHelp HTML project.

Backups? We doan need no steenkin b@(kpppppppp.....____________________________ (End of line)

Even though I preach this time and again, things happen that cause one to be immersed in this distressed state. I am not immune and found myself in such a state once. So even though you may *THINK* you have a decent backup plan, things can sometimes go awry and foil your best laid plans and efforts.

Personally, I discovered what worked best for my situation was to use WinZip and create a complete copy of my project at the end of each workday. I would add everything except the !SSL! folder, as this is easily re-created once you are back in business with your project. I then burned the zip file to CD-ROM, using the date as a file name. For example: **04232004Backup.zip**. I would simply add the newest zip file each day until I filled the CD to capacity.

How to recover in the event of catastrophic failure

Assuming you have a good backup, you should easily be able to restore from it and merrily hum along. However, odds are that you are reading this because the backup failed, or the source files were lost or are somehow otherwise unavailable.

The main thing to do is not panic. Hang in there and we will get you through it. Yes, it likely won't be a cake walk, but I assure you that it can be done, as I've done it myself.

Step 1. Analyze the existing WebHelp project

Odds are that you have a functional version of the WebHelp. This is a very good thing! Your first step is to analyze it to see the folder structure (if any), the Table of Contents structure, the Index terms used, the Glossary terms used and any Browse sequences that have been defined. Your goal will be to duplicate these structures and terms in a new RoboHelp HTML project. **The main thing you need to remember is NOT to make any changes to this version until you are absolutely certain you have successfully duplicated the project!**

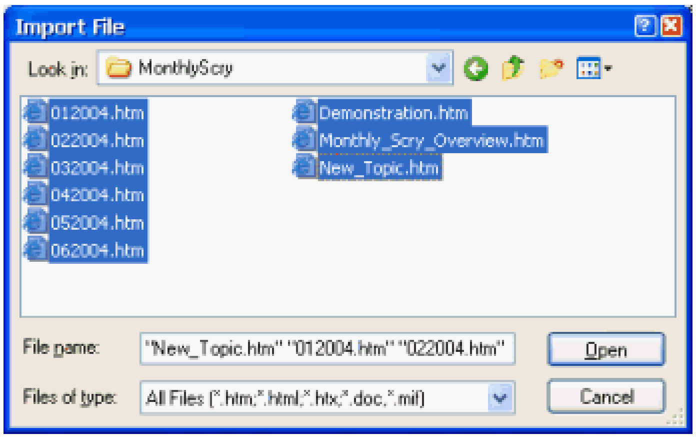
Step 2. Create a new RoboHelp HTML project

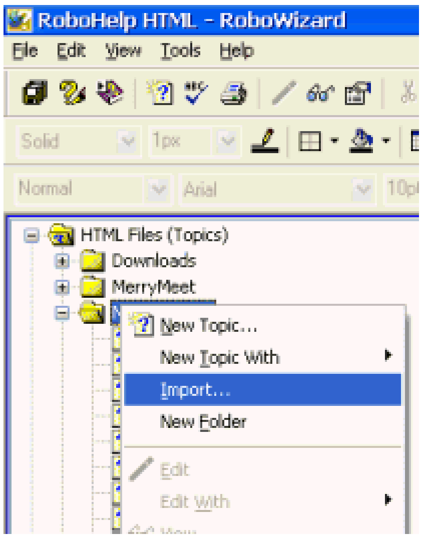
Your goal here is to duplicate the folder structure (if any) of the existing WebHelp project.

Step 3. Add the existing WebHelp files back into the new RoboHelp HTML project

Yes, I'm all too aware that these are WebHelp Output files that have all sorts of extra code in them. And yes, if you were to simply import these and run with them, lots of other issues would result. However, as we have no backup source files, this will help keep from re-creating the wheel.

Basically, you right click each folder and select **Import** from the context menu. Then select the files for the folder and click the **Open** button.


Import File
Look in: MonthlyScry
012004.htm
022004.htm
032004.htm
042004.htm
052004.htm
062004.htm
Demonstration.htm
Monthly_Scry_Overview.htm
New_Topic.htm
File name: "New_Topic.htm" "012004.htm" "022004.htm"
Files of type: All Files (*.htm;*.html;*.htx;*.doc;*.mif)
Open
Cancel

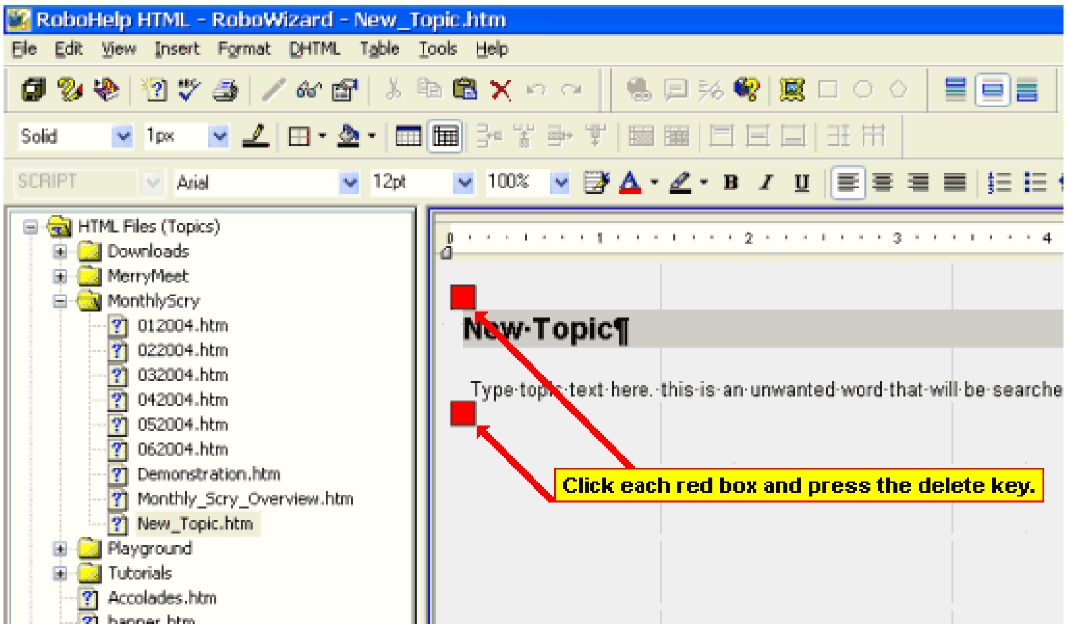


*NOTE: **At various points along the way you may encounter a dialog advising that one or more files already exists and asking if you want to overwrite the file(s). Typically this is the style sheet or possibly a JavaScript file. Either way, you are probably safe with either choice.**

Step 4. Separating the Wheat from the Chaff

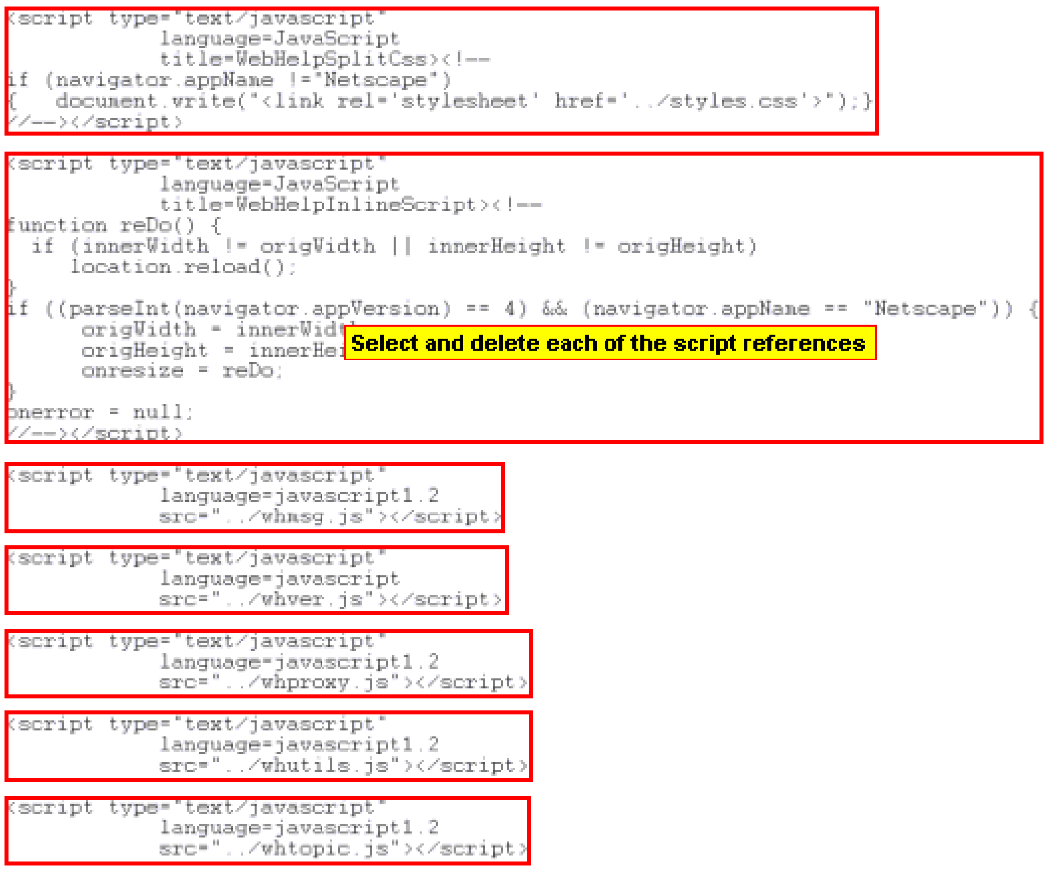
Okay, look at all the progress we have made! Looks like you are almost home. But don't think we are all *THAT* close. There is still much to be done. Once you have imported the WebHelp topics into the source system, your task is to eradicate all the unwanted JavaScript code. You may find yourself wondering why this is necessary. This is because when WebHelp is generated, each source topic is massaged a bit. A small amount of

JavaScript code is added that helps the WebHelp output file survive in multiple browser environments. So the first step is to open each topic in the RoboHelp HTML WYSIWYG editor and delete all the red boxes you see.


RoboHelp HTML - RoboWizard - New_Topic.htm
File Edit View Insert Format DHTML Table Tools Help
Solid 1px
SCRIPT Arial 12pt 100%
HTML Files (Topics)
Downloads
MerryMeet
MonthlyScry
012004.htm
022004.htm
032004.htm
042004.htm
052004.htm
062004.htm
Demonstration.htm
Monthly_Scry_Overview.htm
New_Topic.htm
Playground
Tutorials
Accolades.htm
New·Topic¶
Type·topic·text·here.·this·is·an·unwanted·word·that·will·be·searche
Click each red box and press the delete key.

Step 5. Using the fine-toothed comb to strip the scripts

Now that you have the easy part out of the way, you still have a bit of work ahead of you. Step 4 simply removed the most easily found and obvious code. Your goal now is to visit the TrueCode window and scrub the remaining bits of code. Basically you are on a search and destroy mission. Look for anything beginning with **<script>**. Select from the **<script>** to the closing **</script>** tags and delete them. Most of these will be found in the Head area of the HTML document. So you may need to scroll up a tad to find them all. In my example, there are seven additional script references to remove.



```
<script type="text/javascript"
            language=JavaScript
            title=WebHelpSplitCss><!--
if (navigator.appName !="Netscape")
{   document.write("<link rel='stylesheet' href='../styles.css'>");}
//--></script>
```

```
<script type="text/javascript"
            language=JavaScript
            title=WebHelpInlineScript><!--
function reDo() {
  if (innerWidth != origWidth || innerHeight != origHeight)
     location.reload();
}
if ((parseInt(navigator.appVersion) == 4) && (navigator.appName == "Netscape")) {
      origWidth = innerWid
      origHeight = innerHe
      onresize = reDo;
}
onerror = null;
//--></script>
```

```
<script type="text/javascript"
            language=javascript1.2
            src="../whmsg.js"></script>
```

```
<script type="text/javascript"
            language=javascript
            src="../whver.js"></script>
```

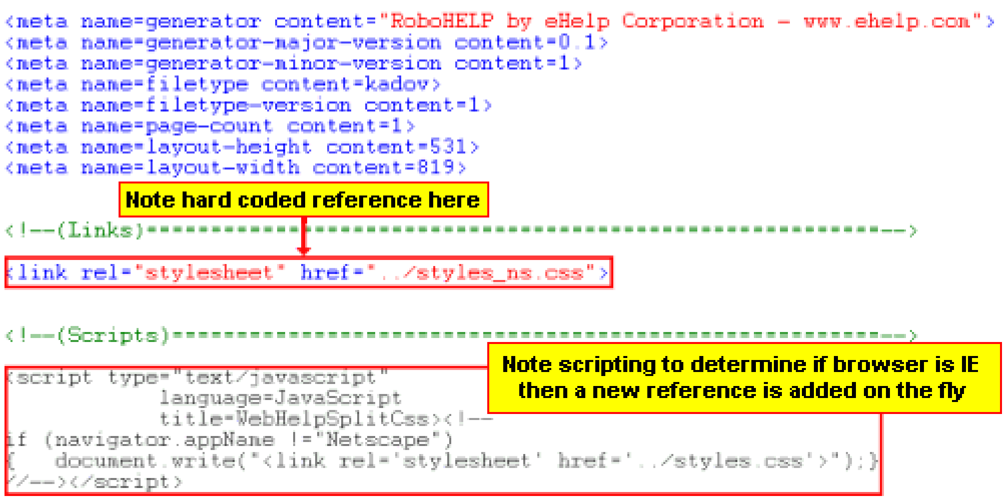
```
<script type="text/javascript"
            language=javascript1.2
            src="../whproxy.js"></script>
```

```
<script type="text/javascript"
            language=javascript1.2
            src="../whutils.js"></script>
```

```
<script type="text/javascript"
            language=javascript1.2
            src="../whtopic.js"></script>
```

Step 6. Re-associating the style sheet

When you were stripping the scripts in Step 5, one script had a reference to your style sheet. Actually, in my example, it was the first script on the page. Notice that a reference was set initially to my **stylesheet name_ns.css**. This is a slightly modified version of my "real" style sheet. My example uses a style sheet bearing the name of "**styles.css**". So the initial hard coded reference is to **styles_ns.css**. So your goal is to ensure your actual style sheet is part of the project, then change the reference from the **_ns.css** to just the style sheet name. Realistically, RoboHelp HTML makes assigning style sheets so easy within the application, that it's probably best to ignore the reference in TrueCode, then re-associate the style sheet from within the RoboHelp HTML application. This step is present simply to point out that the style sheet reference needs changing, and why.


Note hard coded reference here
Note scripting to determine if browser is IE then a new reference is added on the fly

```
<meta name=generator content="RoboHELP by eHelp Corporation - www.ehelp.com">
<meta name=generator-major-version content=0.1>
<meta name=generator-minor-version content=1>
<meta name=filetype content=kadov>
<meta name=filetype-version content=1>
<meta name=page-count content=1>
<meta name=layout-height content=531>
<meta name=layout-width content=819>

<!--(Links)=============================================================-->

<link rel="stylesheet" href="../styles_ns.css">

<!--(Scripts)===========================================================-->

<script type="text/javascript"
            language=JavaScript
            title=WebHelpSplitCss><!--
if (navigator.appName !="Netscape")
{   document.write("<link rel='stylesheet' href='../styles.css'>");}
//--></script>
```

Step 7. Re-establish DHTML

Why would we need to re-associate any DHTML? The simple answer is that if your WebHelp topics included any DHTML effects (DHTML Drop Down areas, DHTML Expanding Text, etc.) It's simply easier and more efficient to re-create these than it is to sift through the existing scripting and make them work. This aspect is one of the reasons that you don't want to change anything about the existing WebHelp files until you are certain that everything has been restored.

Step 8. Re-creating the Table of Contents (TOC), Index, and Glossary

Again, as I stated above, this aspect is another of the reasons that you don't want to change anything about the existing WebHelp files until you are certain that everything has been restored. You will need to methodically step through each of the TOC, Index, and Glossary items to ensure that everything is mapped satisfactorily.

Step 9. Re-creating the Browse Sequences

Yet again, as stated earlier, this aspect is just another of the reasons that you don't want to change anything about the existing WebHelp files until you are certain that everything has been restored. You will need to methodically step through each browse sequence to ensure that everything is mapped satisfactorily.

Step 10. Re-creating any skins used

Fortunately, this one should be pretty simple. When you generate WebHelp using a skin, all the associated files are present in the WebHelp output. They aren't modified, so it's a simple matter to copy them to the appropriate location in your project.

That wraps it up for this month.

Until next month... Namaste and Merry part!