February 5, 2007

The eLearning Guild's LEARNING SOLUTIONS℠

Practical Applications of Technology for Learning

e-Magazine

THIS WEEK: Development Techniques

Tutorial: Build SCORM-Compatible Lesson Templates for Your LMS

By B.J. Schone

When I began working as the sole e-Learning developer for Ferrellgas, a company with several thousand employees, I realized that my time was going to be stretched thin. I was hired to select, implement, and manage a Learning Management System (LMS), and then to develop e-Learning courses for the LMS. For those who have been in a similar position, you know that this is no small task.

After several months, we implemented the LMS and I was ready to begin developing courses. I decided that I wanted to build some type of a template to speed up course development. I dreaded the idea of duplicating code; I did not want to write and re-write the same code for each course (e.g., SCORM functions, navigation, scoring, etc.). Why should I re-invent the wheel each time I need to build a course?

Following a bit of research, brainstorming, and testing, I developed an e-Learning lesson template, which is essentially a framework I can use to build lessons rapidly. The template contains the basic navigation and SCORM functionality that will make the lesson work in an LMS. I am able

The typical e-Learning project requires so much time, and can be so labor-intensive, that productivity is always a concern of developers, designers, and their managers. In addition, making courses SCORM-compliant and functional with the LMS adds another layer (or two) of complexity. This week's article provides a template that will save time while it allows developers and designers to focus on the most important problem, producing learning.

A publication of THE ELEARNING GUILD™

Extra Insights
Page 11

to drop in content, interactive exercises, and assessment questions, and the template does the rest of the work. The ability to use (and re-use) this framework helps to reduce development time, and allows me to concentrate on a lesson's content and interactivity rather than on its delivery details.

This article outlines the benefits of using a lesson template, and describes how you can build and use one at your organization.

What is required to build a lesson template?

At a minimum, you will need a text editor (for writing code) and a graphic editing program. The ideal toolset would be Dreamweaver, and either Fireworks or Photoshop. Other comparable Web development tools should work just as well. You may choose to use Flash or other development tools when developing the actual lesson content, but these basic tools are sufficient to create the lesson template.

You will also need a solid grasp of HTML, and at least some working knowledge of JavaScript and Cascading Style Sheets (CSS). You will need to be familiar with SCORM (or AICC, depending on your situation). Minor graphic editing skills will be helpful, too. Do not worry if you are lacking some of these skills. I've listed several references throughout the article to help you get up to speed.

Finally, you will need some code resources. I recommend the Web sites listed in Sidebar 1 on page 3, and I'll be referring to these throughout the article as I show you how to build a lesson template.

Structure of a course

At our organization, courses consist of one or more lessons. In SCORM-speak, each lesson is its own Sharable Content Object (SCO). You will see that the lesson template is really a SCO template. This means that each lesson can be uploaded to a SCORM-compliant LMS, and it can function as a stand-alone SCO. There are three types of lessons: completion-based, score-based, and assessment-only.

Completion-based lessons

Think of these lessons as "FYI" for the learner, almost like browsing a brochure. For example, we created a course at Ferrellgas to assist employees with their annual medical and dental benefits enrollment. This course contained one lesson, and we offered it for informational purposes only. Employees were not

At a minimum, you will need a text editor (for writing code) and a graphic editing program. The ideal toolset would be Dreamweaver, and either Fireworks or Photoshop. … You may choose to use Flash or other development tools when developing the actual lesson content … .

The eLearning Guild's LEARNING SOLUTIONS℠ Practical Applications of Technology for Learning e-Magazine

Publisher David Holcombe

Editorial Director Heidi Fisk
Editor Bill Brandon
Copy Editor Charles Holcombe

Design Director Nancy Marland Wolinski

Learning Solutions e-Magazine™ is published weekly for members of ***The eLearning Guild***, 375 E Street, Suite 200, Santa Rosa, CA 95404. Phone: +1.707.566.8990. www.eLearningGuild.com

required to take it. Our goal in this case was to distribute information to people who were interested. If the learner paged through the lesson, they received credit for taking it. With this type of lesson, you can either give 100% credit to the learner if they open and browse it, or you can score them incrementally. For example, if the learner visits 7 out of 10 pages in a lesson, they would receive a 70%. Choose the most appropriate scoring method based on your lesson.

This article describes how to build a completion-based lesson. With additional work you can set up your lesson template to support all three lesson types using the information here and in *In the Eye of the SCORM* (reminder: see Sidebar 1).

Score-based lessons

In a score-based lesson, the learner works through several pages of content, interactive exercises, and assessment items. JavaScript in the template tracks the learner's score throughout the lesson, and then reports it back to the LMS at the end (or when the learner closes the lesson window). For more information on score-based lessons, read the "Relating score and success status" section of Chapter 8, *In the Eye of the SCORM*.

Assessment-only lessons

Assessment-only lessons are quizzes or exams that usually come after one or more completion- or score-based lessons. There are several options to consider when building assessment lessons. If you use the lesson template, your assessments will have the same look-and-feel as the rest of your lessons, which is a good thing. Some assessment questions need to be highly customized, and are not the typical multiple-choice, true/false variety. In these cases, we use tools such as Fireworks and Flash to build the assessment, and then deliver it using the lesson template. An example of this would be an assessment that required the learner to perform some type of procedure, given a particular scenario.

For assessments with typical question types (e.g., multiple-choice, true/false, etc.), we use third-party products to speed up development. We use Respondus StudyMate (http://www.respondus.com/products/studymate.shtml) for short quizzes, or Atrixware's Test Pro Developer (http://www.atrixware.com/Web8/products/developer.php) for longer quizzes and exams. We especially like Test Pro Developer because it allows us to track learners' responses down to the individual question level. The ability to view this level of detail is important for Ferrellgas' safety and compliance training. Both StudyMate and Test Pro Developer output their quizzes and exams as a SCORM package, which you can upload directly to an LMS.

How to build a lesson template

At first glance, this template will appear to be a Frankenstein monster pieced together from random parts. The end goal is to have all of the pieces working in unison so that the lesson works smoothly, both behind the scenes and for the learner.

Getting started

Before I begin showing you how to build a template, there are six key elements in my approach to be aware of: SCORM bias, audience awareness, standards, accessibility, portability, and persistence.

SCORM bias

The majority of this article focuses on the use of the SCORM 2004 communication and tracking, as opposed to AICC. This is because SCORM 2004 is one of the most widely used methods of communication between courses and LMSs, and our LMS supports SCORM 2004. You should use the method of communication that applies to your organization's LMS. You will find that many of the principles in this article will still work when used in conjunction with AICC or SCORM 1.2; however, you will need to make some code changes.

I developed an e-Learning lesson template, which is essentially a framework I can use to build lessons rapidly. The template contains the basic navigation and SCORM functionality that will make the lesson work in an LMS. I am able to drop in content, interactive exercises, and assessment questions, and the template does the rest of the work.

Sidebar 1: *Recommended resources*

JavaScript Toolbox (http://www.javascripttoolbox.com)

This is a great code repository and reference site for the JavaScript language. I've used the DHTML Tree code from this site to build the lesson template.

Ostyn Consulting – Resources (http://ostyn.com/resources.htm)

Claude Ostyn, one of the leading experts on SCORM, has written an e-Book called *In the Eye of the SCORM: An introduction to SCORM 2004 for Content Developers*. This e-Book is a must-have for anybody working with SCORM and it is free on the Ostyn Consulting Web site. In this tutorial, I use several SCORM JavaScript functions from Chapter 8 ("Practical SCO Construction") in the lesson template. There are other articles, references, and tools available on the Ostyn Consulting Web site that may also be useful to you later as you develop your own templates.

Make sure you read and abide by the license agreements set forth by each of these Web sites regarding the use of their code. Also note that *In the Eye of the SCORM* is currently a Draft. Please send feedback to Ostyn Consulting if you encounter a bug in the code. Be aware that Claude updates and makes minor revisions to the e-Book fairly often, so specific items that I refer to in this article may change location or titles, and may even drop out of the book. Please contact me, not Claude, if you are unable to locate a particular reference (bjschone@gmail.com). All references were current at the time of this article's publication.

Know your audience

Make sure to design your lesson template to work well for your audience, from a technical and usability perspective. Keep the following questions in mind: Are audience members using broadband or dial-up Internet access? What Web browsers do they use and which versions? What screen resolution do they prefer – and do your lessons function correctly when viewed at the same resolution? What plug-ins do they have (or not have)? Test your lessons as if you are the user. Use your QA department, if you have one. Ask some of the users if they would be willing to test some of the lessons for you. Design for their needs and solicit their feedback; this is essential for success in e-Learning.

Set standards

I highly recommend that you establish design and development standards with your e-Learning developers. Consistent and disciplined use of naming conventions, text formatting, CSS, and page layout can save hours of time (and headaches) in the long run. For more information on creating design and development standards within your organization, see Mike Dickinson's excellent articles on creating and using an e-Learning developer's guide in the September 25, 2006, and October 16, 2006, editions of *Learning Solutions e-Magazine*.

Be accessible

It is a best practice to build your e-Learning to be accessible to those with disabilities. For more information on accessibility, visit the Section 508 Web site (http://www.section508.gov/) or the Web Accessibility Initiative (WAI) Web site (http://www.w3.org/WAI/). You may also want to refer to Amy Kellogg's article on Section 508 compliance and "hidden disabilities" in the October 10, 2005 *Learning Solutions e-Magazine*. Martie Buzzard also published more general articles on e-Learning and accessibility in the October 11, 2004 and October 8, 2002 issues.

Portable lessons

Keep in mind that it may often be useful to have lessons stand alone, outside of an LMS. Take this scenario: Your manager says, "I need to demo our newest PeopleSoft course to show our Vice President today at lunch. I'll have my laptop, but won't have Internet access. Can you give the course to me on CD?" This lesson template will allow you to access courses outside of an LMS. You can copy the lesson's folders to a CD, and your manager will be ready to go. But keep in mind that scoring and tracking functionality will not be available for lessons that are launched on their own.

Persisting user information

It takes some fancy footwork in order save user information (e.g., scores, learner progress, etc.) as a learner moves through an e-Learning lesson. This is due to the nature of HTML Web pages; the pages cannot "remember" user information as the individual moves from one page to another. One solution for overcoming this problem is to use an HTML frameset. When using a frameset, the learner will only be changing pages in one of the frames, often called the content or display frame (see Figure 1). The other frames do not change, so we can store the user's information in these static frames. For more information on this issue, see the section titled "Maintaining state across multiple pages" in Chapter 8 of *In the Eye of the SCORM*.

Some people may think I am crazy for recommending the use of frames because it is considered an

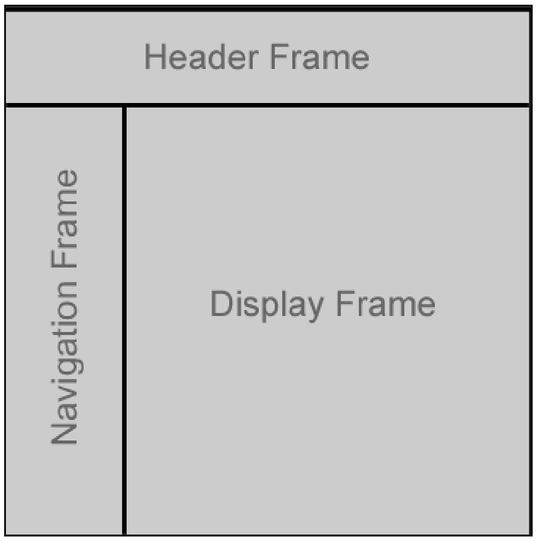


Figure 1 ***A frameset gives the developer a place to store user information – in the static frames.***

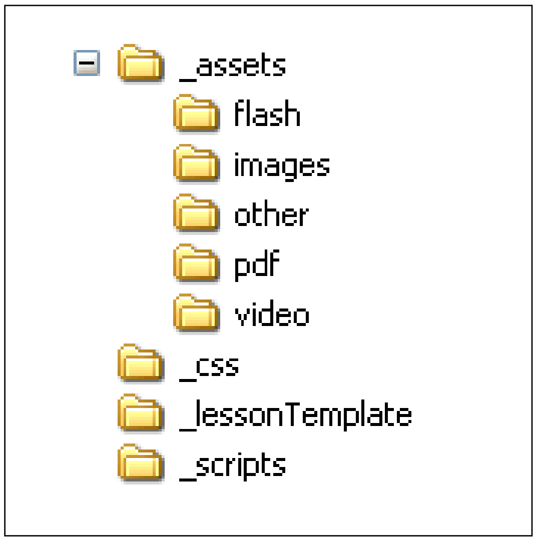


Figure 2 ***Your lesson template folders should have this organization structure.***

antiquated Web development practice. But I assure you it does work well, and we have received great feedback from our learners regarding our courses. If you are hesitant to go with the traditional frameset, you may consider attempting a more modern layout by using a hidden, or very small, frame to retain learner information.

Building the template

Building the template is a step-by-step process. There is a lot of detail to attend to, and I'll walk you through all of it in this section of this article.

Step 1. Create the lesson template folders

First, create a new folder somewhere on your PC for the lesson template. The title of this folder is not important. Inside your new folder, create the following subfolders: `_assets`, `_css`, `_lessonTemplate`, and `_scripts`.

Next, in the `_assets` folder, add subfolders named `flash`, `images`, `other`, `pdf`, and `video`. The main idea of the `_assets` folder and its subfolders is to hold files that are used across all lessons. For example, you may want to keep a copy of your company's logo in the `_assets\images` folder, as it is something you would probably have displayed in the header of all of your lessons. Your folders should now look like Figure 2 on page 4.

The idea of this folder structure is to centralize the main elements of the template and keep them separate from each lesson's content. For example, you will only have one `_scripts` folder and it will contain the JavaScript files used to operate the template. So if a JavaScript bug pops up that affects all of your lessons, you can fix it in this one place instead of having to fix it for each lesson. This method has provided us with a simple way to keep track of our files, and, if we need to make changes, it is quick and easy to repackage and republish the lessons to the LMS. The more you use this file structure, the more sense it will make.

Let me say that last bit again: If you experience a JavaScript bug or some other glitch, even though you only need to fix it in one place, you will still have to repackage and republish your lessons to your LMS. This is not difficult, but it can take a bit of time. It still beats the alternative: fixing the bug in multiple places and then having to repackage and republish the lessons.

Step 2. Create lesson-specific folders

Add four subfolders in the `_lessonTemplate` folder: `_content`, `_localAssets`, `_localScripts`, and `_localCSS`. These folders will hold lesson-specific information.

Next, add these subfolders to the `_localAssets` folder: `flash`, `images`, `other`, `pdf`, and `video`. You may be thinking, "Didn't I just add these a few minutes ago?" Well, sort of. There are two sets of folders for maintaining our lesson assets. The `_assets` folder is the higher-level folder that holds items that are used across all lessons. The `_localAssets` folder holds items that are specific to one particular lesson. If you create a lesson on building widgets, you would have images representing the different types of widgets in the `_localAssets\images` folder.

Your folders will end up looking like Figure 3.

Step 3. Set up files

Next, use your text editor, or Dreamweaver, to create the following files. Leave each file empty unless otherwise specified.

- In the `_lessonTemplate` folder, create a file called `index.html`.
- In the `_lessonTemplate` folder, create a file called `headerFrame.html`. In this file, add the basic tags used in an HTML document (<html>, <head>, <body>, etc.). You do not need to add anything else to this file at this time.
- In the `_lessonTemplate` folder, create a file called `navFrame.html`. In this file, add the basic tags used in an HTML document (<html>, <head>, <body>, etc.). You do not need to add

There are three types of lessons: completion-based, score-based, and assessment-only. … Think of completion-based lessons as "FYI" for the learner, almost like browsing a brochure. … In a score-based lesson, the learner works through several pages of content, interactive exercises, and assessment items. … Assessment-only lessons are quizzes or exams that usually come after one or more completion- or score-based lessons.

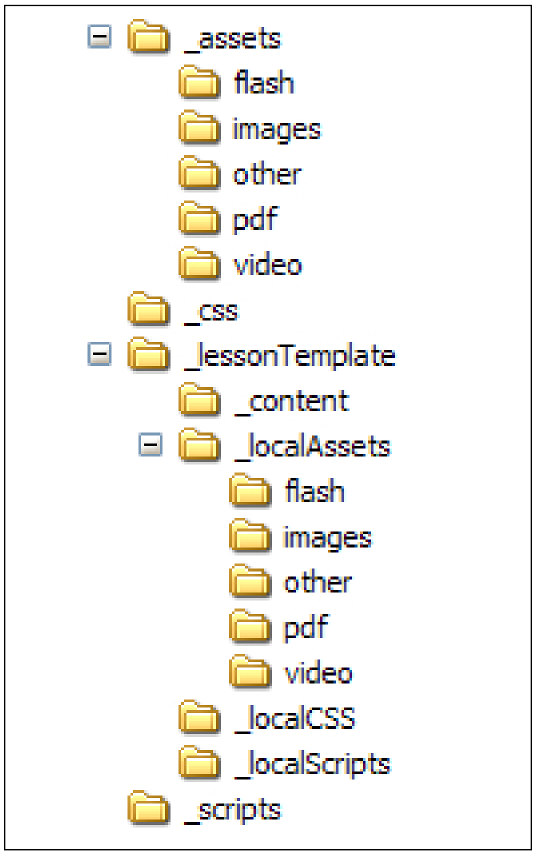

```
_assets
    flash
    images
    other
    pdf
    video
_css
_lessonTemplate
    _content
    _localAssets
        flash
        images
        other
        pdf
        video
    _localCSS
    _localScripts
_scripts
```

Figure 3 *With lesson-specific folders added, your folder structure will look like this.*

anything else to this file at this time.

- In the `_css` folder, create a file called `styles.css`. CSS allows you to control the layout and text formatting for your Web site – or in this case, your lesson – from one location. If you have a standard set of CSS styles at your organization, put them in this file. Otherwise, add your own styles to this style sheet as you build and refine your template. (If you are new to Cascading Style Sheets, I recommend *Cascading Style Sheets: The Definitive Guide* by Eric Meyer, an excellent reference for all levels of CSS expertise.)
- In the `_scripts` folder, create a file called `global.js`. You will use this file to hold functions that will be accessed throughout the entire template, such as string functions, scoring functions, and more. Think of this file as a place to put JavaScript functions that all (or most) lessons will use.
- In the `_scripts` folder, create another file called `simpleSCO.js`. This file will hold a majority of the SCORM functionality required for interacting with the LMS.
- In the `_scripts` folder, create a third file called `navTree.js`. This file will hold the DHTML Tree functions that allow your navigation tree to expand and collapse.
- In the `_lessonTemplate\_localScripts` folder, create a file called `lessonParameters.js`. This file will hold lesson-specific information, such as the lesson title, mastery score, etc. Add this line of code to this file:

```
var CONST_lessonName="LESSON TITLE
  GOES HERE";
```

Step 4. Build the frameset

Next, I'll show you how to build a frameset that has a header frame, a navigation frame, and a content frame, as you saw in Figure 1 earlier. Enter the code seen in Figure 4 into your `index.html` file.

Feel free to experiment with different frameset layouts once you are comfortable with how the lesson template is structured. You may find that another layout better suits your needs.

I used the Ferrellgas logo and company colors very prominently when designing the header and navigation frame of our lesson template (see Figure 5 on page 7). You may consider using your organization's logo and/or color scheme, too. Learners will become familiar with the look-and-feel over time, and your lessons will be consistent with your organization's marketing style.

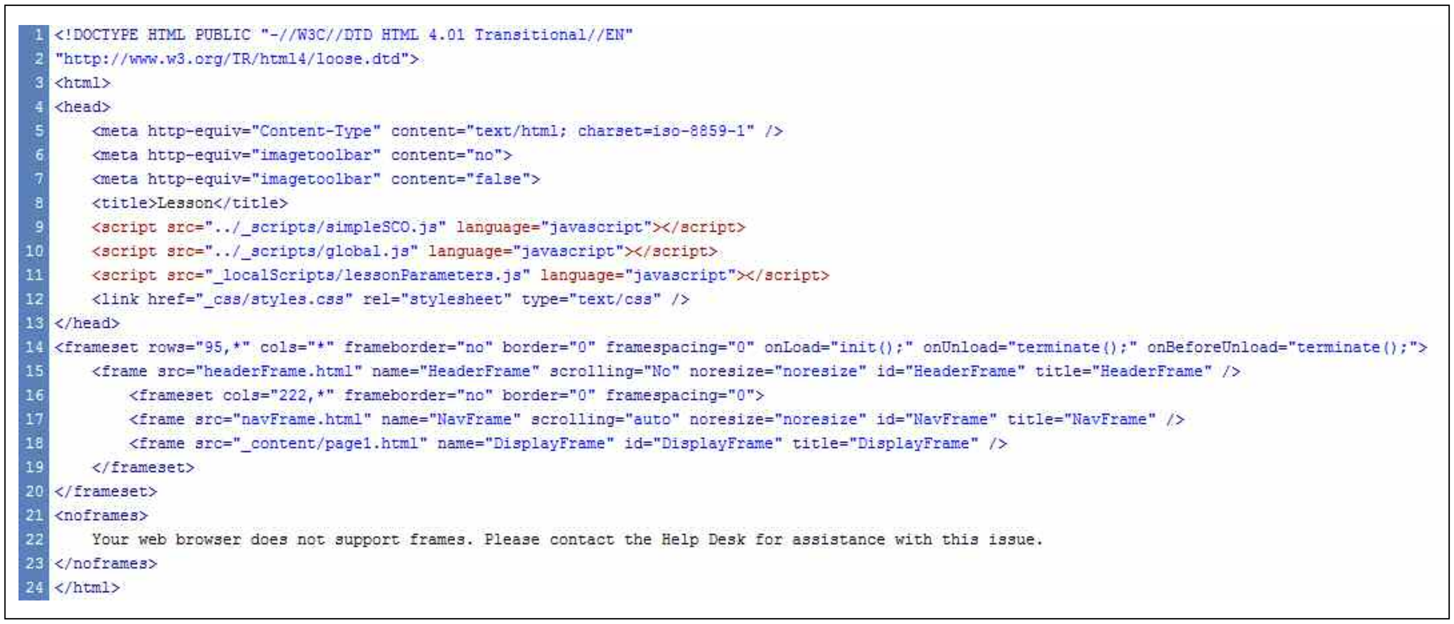
Add a navigation system

Next, you must add a navigation system to allow the learner to move between the pages of the lesson. This navigation system comes from a DHTML script, called the DHTML Tree.

To obtain the code for the DHTML Tree, visit the JavaScript Toolbox Web site (http://www.javascripttoolbox.com) and click on the DHTML Tree link. Once there, copy the source code for the DHTML Tree (open the tab on the site named "Source" and you will see the code in a gray text box) and paste it into the `_scripts\navTree.js` file. (Note: You have the choice of copying either the fully commented code or the compacted code; they work the same way.) Next, include this file in the `navFrame.html` file using the code below.

I highly recommend that you establish design and development standards with your e-Learning developers. Consistent and disciplined use of naming conventions, text formatting, CSS, and page layout can save hours of time (and headaches) in the long run.

Figure 4 *This code adds the frameset seen in Figure 1 to your template.*



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <meta http-equiv="imagetoolbar" content="no">
    <meta http-equiv="imagetoolbar" content="false">
    <title>Lesson</title>
    <script src="../_scripts/simpleSCO.js" language="javascript"></script>
    <script src="../_scripts/global.js" language="javascript"></script>
    <script src="_localScripts/lessonParameters.js" language="javascript"></script>
    <link href="_css/styles.css" rel="stylesheet" type="text/css" />
</head>
<frameset rows="95,*" cols="*" frameborder="no" border="0" framespacing="0" onLoad="init();" onUnload="terminate();" onBeforeUnload="terminate();">
    <frame src="headerFrame.html" name="HeaderFrame" scrolling="No" noresize="noresize" id="HeaderFrame" title="HeaderFrame" />
        <frameset cols="222,*" frameborder="no" border="0" framespacing="0">
        <frame src="navFrame.html" name="NavFrame" scrolling="auto" noresize="noresize" id="NavFrame" title="NavFrame" />
        <frame src="_content/page1.html" name="DisplayFrame" id="DisplayFrame" title="DisplayFrame" />
    </frameset>
</frameset>
<noframes>
    Your web browser does not support frames. Please contact the Help Desk for assistance with this issue.
</noframes>
</html>
```

```
<script src="../_scripts/navTree.js"
  language="javascript"></script>
```

You will also need to download the following files for the DHTML Tree (right click on the "Other Required Files" links on the Source tab): `mktree.css`, `plus.gif`, `minus.gif`, and `bullet.gif`.

Place the three images (`plus.gif`, `minus.gif`, and `bullet.gif`) in the `_assets\images` folder. Next, copy the CSS out of the `mktree.css` file and paste it in the `_css\styles.css` file. In this CSS, edit the three paths of the image files (e.g., change `minus.gif` to `../_assets/images/minus.gif`, etc.)

You will not need to add any links to the navigation tree at this point; however, you can refer to Figure 5 to see what the navigation tree will look like in the final product.

Step 6. Set up the header frame

In the `headerFrame.html` file, add this code to display the lesson title throughout the lesson and in the window title:

```
<script language="javascript">
  document.write(top.CONST_lessonName);
  top.window.document.title=top.CONST_
    lessonName;
</script>
```

This causes the "Introduction to FerrellConnect" title text to display in Figure 5.

Step 7. Specify LMS communication method

The method of communication between your lessons and your LMS will most likely be SCORM 1.2, SCORM 2004, or AICC. From *In the Eye of the SCORM*, use the code in the Chapter 8 sections "A more complete reusable SCO script" and "Multi-page SCO using the generic script." This code, compatible with SCORM 2004, is straightforward and easy to implement. Follow these steps to set up the code.

Place these JavaScript functions from the e-Book into the `simpleSCO.js` file in the `_scripts` directory: `ScanForAPI()`, `GetAPI()`, `ScormInitialize()`, `ScormTerminate()`, `ScormGetLastError()`, `ScormGetErrorString()`, `ScormGetValue()`, and `ScormSetValue()`.

Place the following JavaScript functions from the e-Book into the `global.js` file in the `_scripts` directory: `GoToPage()`, `GoPreviousPage()`, `GoNextPage()`, and `MarkIfCompleted()`.

Copy the following variables from the e-Book into the `lessonParameters.js` file in the `_localScripts` folder: `gnPage`, `gnMaxPages`, `gnPagesNeeded`, `gaPagesCompleted`, and the corresponding `for{}` loop.

You will need to make one small modification to the `GoToPage()` function in the `global.js` file. Add the folder name `_content` to this line to provide the correct path to your content files:

```
DisplayFrame.location.href =
  "_content/page" + gnPage + ".html";
```

Customize your frames

Add two buttons in the `headerFrame.html` – one that says "Previous," and one that says, "Next." We then add `onClick` JavaScript events to make these buttons work. The code is as follows:

```
<form name="headerForm"id="headerForm">
  <input type="button"name="prevButton"
     id="prevButton" value="Previous"
     onClick="top.GoPreviousPage()" />
  <input type="button"
     name="nextButton" id="nextButton"
     value="Next" onClick="top.
     GoNextPage()" />
</form>
```

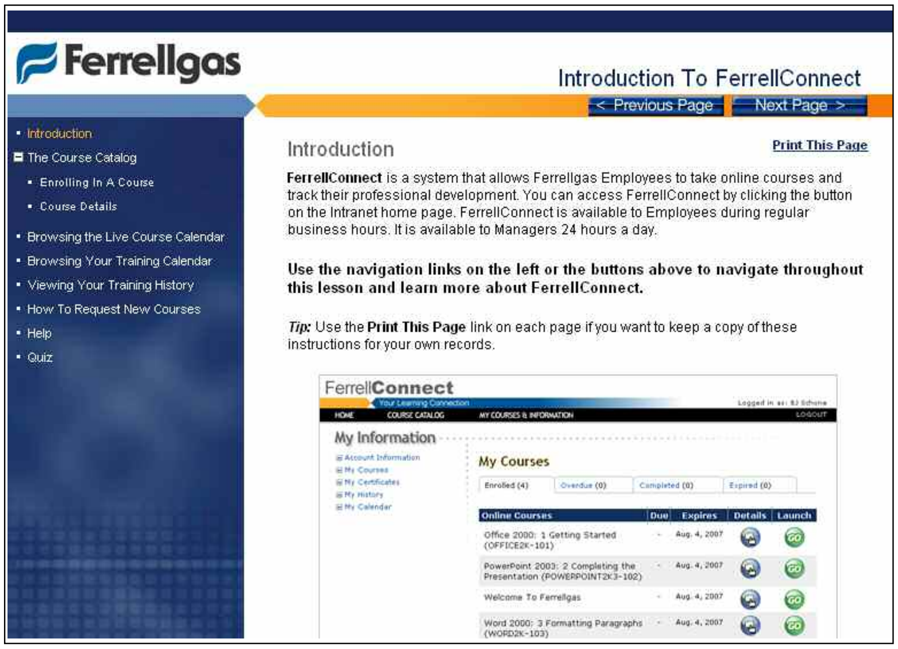
You may also choose to add tables or other formatting to your `headerFrame.html` page to organize these buttons, your organization's logo, etc.

How to use a lesson template

Follow the steps below to create a lesson and publish it to your LMS. These steps assume that you have already worked through the instructional design

If you experience a JavaScript bug or some other glitch, even though you only need to fix it in one place, you will still have to repackage and republish your lessons to your LMS. This is not difficult, but it can take a bit of time.

Figure 5 ***In use, your frameset should look something like this.***



Introduction

FerrellConnect is a system that allows Ferrellgas Employees to take online courses and track their professional development. You can access FerrellConnect by clicking the button on the Intranet home page. FerrellConnect is available to Employees during regular business hours. It is available to Managers 24 hours a day.

Use the navigation links on the left or the buttons above to navigate throughout this lesson and learn more about FerrellConnect.

Tip: Use the **Print This Page** link on each page if you want to keep a copy of these instructions for your own records.

Online Courses	Due	Expires	Details	Launch
Office 2000: 1 Getting Started (OFFICE2K-101)	-	Aug. 4, 2007		GO
PowerPoint 2003: 2 Completing the Presentation (POWERPOINT2K3-102)	-	Aug. 4, 2007		GO
Welcome To Ferrellgas	-	Aug. 4, 2007		GO
Word 2000: 3 Formatting Paragraphs (WORD2K-103)	-	Aug. 4, 2007		GO

process to prepare your content.

Make a copy of the `_lessonTemplate` folder. Rename the folder to reflect the new lesson you are creating (e.g., `IntroductionToPeopleSoft`). This new folder must remain at the same directory level as the `_lessonTemplate` folder. If you move it, the lesson template will not function properly. See Figure 6 to see where to locate the folder. You will see that the lesson template folder names are preceded with an underscore, so that they can easily be distinguished from your lesson folders.

In this new folder, open the `_localScripts\lessonParameters.js` file and edit the appropriate parameters (e.g., lesson title, number of pages in the lesson, etc.). With these variables populated, you can now reference their values throughout the lesson.

Create the lesson's content pages in the `<Your LessonNameHere>\_content` folder. Remember to name these pages sequentially based on their position within the lesson (e.g., `page1.html, page2.html`, etc.).

Your content pages may contain simple text, exercises built using JavaScript or Flash, or even video. This is your time to shine! Make the lesson as interactive and interesting as possible, while still adhering to the course and lesson objectives. Store the lesson's assets in the appropriate folders (e.g., `_localAssets\images, _localAssets\flash`, etc.).

Open the `navFrame.html` file and enter the links to each of the individual pages within the lesson. If you used the DHTML Tree code from the JavaScript Toolbox Web site mentioned above, this is as easy as adding an unordered list of links where each link references the `GoToPage()` function. (See code in Sidebar 2.) Note that the `<ul>` tag must have the `CLASS` and `ID` values set for the DHTML Tree to function.

The value you give to the `GoToPage()` function corresponds to the name of the content page. For example, the function call `GoToPage(4)` will take you to `page4.html`.

Save all of the files and then open the `<Your LessonNameHere>\index.html` file to preview the lesson.

Create the SCORM manifest file `(imsmanifest.xml)` in the lesson's folder. You need to create this file because each lesson is its own SCORM package. If you are comfortable with XML, you can manually create this file. If you are not comfortable with XML, several applications can help you to do this. Search the Web for "SCORM Manifest Generator" or "SCORM Manifest Maker" and you will find an application to assist you.

Use a stand-alone SCORM wrapper to test your SCO and verify that it is working correctly. Make sure all appropriate SCORM information is sent back from the SCO to the tester, which shows what information your LMS would receive.

An excellent SCORM wrapper can be found in the SCORM – Technical section on Ostyn.com's Resources page. This wrapper is listed under "SCORM Testing, Demonstration, and Diagnostic Tools," titled "SCO test wrap to monitor SCO Communication with LMS."

Zip up all of the lesson files and upload them to your LMS. You will need to zip up the following files and folders:

- `_assets` folder
- `_css` folder
- `_script` folder
- `<YourLessonNameHere>` folder
- `imsmanifest.xml` file

Do not zip up the `_lessonTemplate` folder; this folder is used only as a template.

Note: You will have to temporarily move your `imsmanifest.xml` file outside of the `<YourLesson NameHere>` folder in order to zip it up in this format. Once you are finished zipping the file up, move it

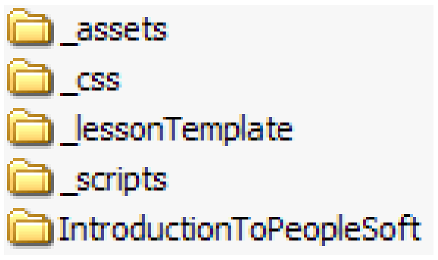


Figure 6 ***Locate the*** `_lessonTemplate` ***folder as shown.***

Sidebar 2: *Unordered list of links*

```
<ul class="mktree" id="tree1">
  <li id="link1"><a href=" javascript:top.GoToPage(1);">
    Introduction</a></li>
  <li id="link2"><a href=" javascript:top.GoToPage(2);">
    Objectives</a></li>
  <li id="link3"><a href=" javascript:top.GoToPage(3);">
    My third page</a></li>
  <li id="link4"><a href=" javascript:top.GoToPage(4);">
    My fourth page</a></li>
  <li id="link5"><a href=" javascript:top.GoToPage(5);">
    My fifth page</a></li>
</ul>
```

back to its folder to avoid any confusion in the future.

I recommend that you give the zip file the same name as the lesson folder. So if the lesson folder is named `PreventingWorkplaceAccidents`, the zip file would be named `PreventingWorkplaceAccidents.zip`.

Release the lesson to your learners once you are confident that everything is working properly.

Suspending and resuming learner progress

You can use the `SaveSuspendData()` and `RestoreFromSuspend()` functions from Chapter 8 of *In the Eye of the SCORM* if you would like to give learners the ability to pause their lesson and come back at a later time. These functions are located in Code Sample 13 on pages 45 and 46 of the e-Book.

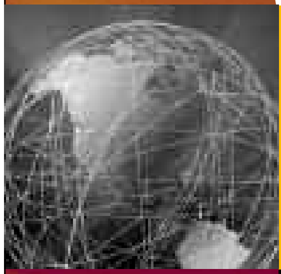
Committing Learner Information

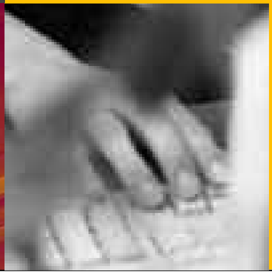
How often should you save the learner's information back to the LMS during the lesson? Every two minutes? Every 20 minutes? Claude Ostyn has written a brief article called "Best practices for the use of Commit in SCORM content" that provides excellent insight. You can find the article in the same SCORM – Technical section as *In the Eye of the SCORM*. You will need to make modifications to the lesson code if you would like to make additional calls to the `ScormCommit()` function.

Closing Thoughts

I would like to offer special thanks to three individuals. Claude Ostyn has made many great contributions to the SCORM community, and I have learned much of what I know by studying his articles and examples. (For the record, I have no affiliation with Claude Ostyn or Ostyn Consulting.) Matt Kruse owns JavaScript Toolbox and is the major provider of the code found there. John Robrimi is a friend with whom I have worked on past e-Learning projects, and who has helped me become a better e-Learning developer.

With their help, I have presented one possible approach for structuring your lesson template. The format and layout of your template will differ depending on your situation. The main idea is to organize your files and structure your template to promote code re-use. The template in this article may look like a typical "page-turner," but you can easily make the content engaging and interactive in order to keep the learner's interest. Most importantly – use what works best for you and your organization.





References

Meyer, Eric. *Cascading Style Sheets: The Definitive Guide*, Second Edition. O'Reilly. 2004.

Ostyn, Claude. *In the Eye of the SCORM*. Retrieved from http://ostyn.com/resources, January 15, 2007.

Author contact

B.J. Schone is an e-Learning Specialist at Ferrellgas in Liberty, Missouri, where he designs and builds online courses and manages FerrellConnect, the Ferrellgas Learning Management System. B.J. earned his Bachelor of Arts in Computer Science and his Master's of Education from the University of Missouri-Columbia. You can contact him via e-mail or phone: bjschone@gmail.com, 913-226-2845.

Discuss this article in the "Talk Back to the Authors" Forum of Community Connections (http://www.elearningguild.com/community_connections/forum/categories. cfm? catid= 17& entercat=y). You can address your comments to the author(s) of each week's article, or you can make a general comment to other readers.

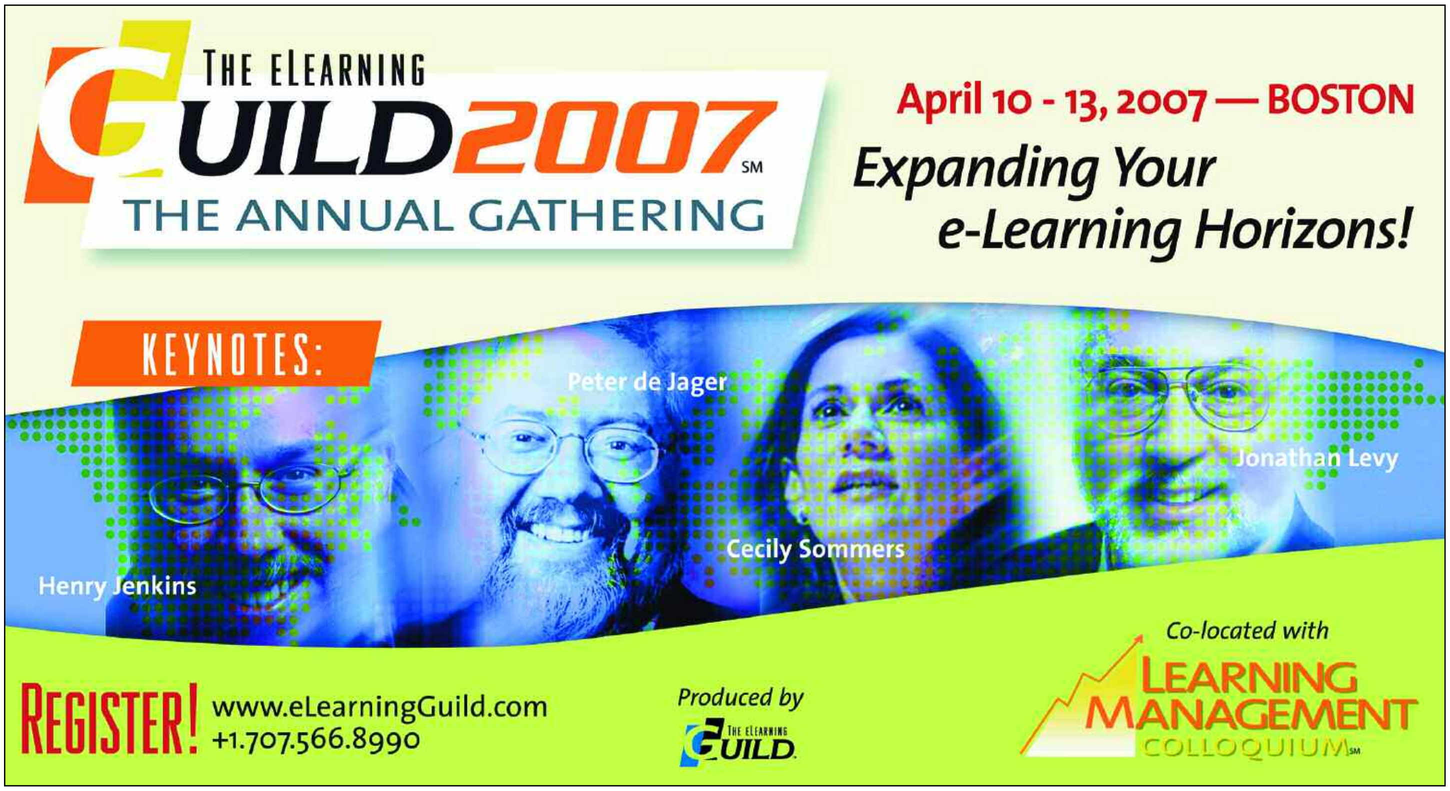
Additional information on the topics covered in this article is also listed in the Guild Resource Directory.

In the Archives

This is B.J. Schone's first article for The eLearning Guild. The eLearning Guild has previously published articles whose topics are related to this week's. These are available to Members in the Learning Solutions Archive online. Members must log in to download them. Here are the authors, the article topics, and the publication dates.

Articles on related topics

Mike Dickinson: Creating and using an e-Learning developer's guide (09/25/06 and 10/16/06)
Kendrick Abell: Templates and reusability (1/23/06)
Monique Donahue: Design documents (12/5/05)
Amy Kellogg: Section 508 compliance and "hidden disabilities" (10/10/05)
Martie Buzzard: e-Learning and accessibility (10/08/02 and 10/11/04)
Joel McKinney: SCORM implementation issues (12/22/03)
Jeffrey Englebrecht: SCORM deployment issues (02/18/03)


THE eLEARNING GUILD2007 SM
THE ANNUAL GATHERING
April 10 - 13, 2007 — BOSTON
Expanding Your e-Learning Horizons!
KEYNOTES:
Peter de Jager
Jonathan Levy
Cecily Sommers
Henry Jenkins
REGISTER! www.eLearningGuild.com +1.707.566.8990
Produced by THE eLEARNING GUILD
Co-located with LEARNING MANAGEMENT COLLOQUIUM SM

Interactive Documents Benefit Designers, Organizations, and Learners

Adobe Acrobat Professional's advanced editing tools make it easy to create documents with interactive elements as simple as rollover text and graphics, or as complex as integrated video and sound. This changes how designers develop materials and how learners use them, and helps organizations move from paper-based materials toward an online learning environment.

As an instructional designer, I have come to rely on Adobe Acrobat to develop materials for both classroom and online instruction that you can view and print using Adobe Reader. Consistent navigation elements ensure easy accessibility to PDF documents, since learners are familiar with pages and bookmarks, and are quite comfortable with viewing and printing support materials. However, for a long time I was satisfied with the Standard version of Adobe Acrobat, and considered the Professional version to be an unnecessary upgrade.

My complacence about Acrobat ended when I became involved in a project that limited learner's access to paper-based training materials. The client wanted to simulate a workplace environment that bans paper due to information security concerns, to limit production and materials costs, and to maintain document integrity through version control. The training plan specified use of interactive PDF documents incorporating button links and rollover hotspots, to encourage in-depth exploration of the content instead of cursory skimming. I would need Adobe Acrobat Professional Version to create the buttons and hotspots.

Being unfamiliar with the software's advanced editing tools, I was apprehensive about using them. However, that apprehension went away, and I'm now a believer in the ease and value of including multimedia content in PDF documents. I can make minor changes to documents with the text touch-up and object touch-up tools – without losing the links I've created. In short, by using Adobe Acrobat Professional I've been able to address the client's stated needs, and have reaped some other unanticipated rewards.

Advantages for the instructional designer

My challenge is to avoid creating materials that are merely printed and filed, and perhaps never used. Instead, I hope to channel the learner's attention to follow the designed lesson path as each activity is completed. I also value any tool that helps me develop content that appeals to a variety of learning styles. Using Acrobat Professional to incorporate multimedia elements definitely fits the bill in these respects. For example, instead of settling for a static diagram, I can include a rollover map for the learner to explore. I can use the link tool to direct the learner to reference materials located elsewhere in the document, in another attached document, or on an external Web site. I can provide audio samples to demonstrate word pronunciation, or visual prompts illustrating a complex process. While exploring the text, graphics, multimedia files, and Internet links, the learner essentially creates a document customized to his own pace, and systematically focuses attention on one element at a time.

Interactive elements also resolve some space issues presented in the static printed page. Rollovers, for example, enlarge the space available for information display, and reduce the need to cut information in order to make everything fit. This makes supplemental information available on demand, without crowding out more critical information.

Advantages for the learning organization

Passing the responsibility for printing learning aids to the learner "on demand" might reduce centralized printing and distribution costs, especially when materials require frequent updates. However, once the learner self-publishes the materials, the organization loses version control of the documents. For example, learners may continue to use "cheat sheets" after the information on them is outdated, putting the organization at risk for the effects of using inaccurate information. It would likely be much better for the organization if the learning aids did not need printing in the first place.

The challenge here is to encourage learners to change the habit of automatically printing documents, and then referring to them when they are no longer current. Materials consisting of interactive content lend themselves to exclusively online use, since it makes little sense to print the documents at all.

Using documents enhanced with interactive elements also fosters the transition from print-based to screen-based learning. The familiarity of text-based documents, paired with readily recognized rollover and button interactions, reduces resistance to change in organizations that are slow to adopt online delivery methods.

Conclusion

Only when I began using Adobe Acrobat Professional's functionality did I start to see the advantages it offered. Gradually, I have recognized how the advances in interactivity affect my design process, and I have noticed that these newly discovered capabilities address some of my longstanding issues, such as space limitations on the printed page, and getting learners to use the materials instead of printing and forgetting them. I would encourage other designers to explore the ways in which interactive documents can enhance online course materials and job aids.

Bonnie Taylor is an Instructional Systems Designer for Karta Technologies, Inc., a training-solutions provider for both the public and private sectors. She creates learning materials for instructor-led and Web-based learning environments, particularly for banking and financial services call centers, and other telecommunications clients. Contact Bonnie by e-mail to btaylor@karta.com.

THE ELEARNING GUILD™

A Worldwide Community of Practice for e-Learning Professionals

The eLearning Guild is a Community of Practice for e-Learning design, development, and management professionals. Through this member-driven community we provide high-quality learning opportunities, networking services, resources, and publications. Members represent a diverse group of managers, directors, and executives focused on training and learning services, as well as e-Learning instructional designers, content developers, Web developers, project managers, contractors, and consultants. Guild members work in a variety of settings including corporate, government, and academic organizations.

Guild membership is an investment in your professional development and in your organization's future success with its e-Learning efforts. Your membership provides you with learning opportunities and resources so that you can increase your knowledge and skills. That's what the Guild is all about ... putting the resources and information you need at your fingertips so you can produce more successful e-Learning.

The eLearning Guild offers four levels of membership. Each level provides members with benefits commensurate with your investment. In the table you will find a comprehensive summary of benefits offered for each membership level. To learn more about Group Membership and pricing, go to www.eLearningGuild.com.

Guild Benefits	Associate	Member	Member+	Premium
eLearning Insider	✓	✓	✓	✓
Annual Salary Survey	✓	✓	✓	✓
Past Conference Handouts	✓	✓	✓	✓
Resource Directory – Access & Post	✓	✓	✓	✓
Info Exchange – Access & Post	✓	✓	✓	✓
Job Board – Access Jobs & Resumes	✓	✓	✓	✓
Job Board – Post Resumes	✓	✓	✓	✓
Job Board – Post Jobs	✗	✓	✓	✓
Guild Research – Online Briefings	✓	✓	✓	✓
Guild Research – Reports	✗*	✓	✓	✓
Guild Research – Archives	✗	✓	✓	✓
Learning Solutions e-Magazine	✗*	✓	✓	✓
Online Forums – Archive	✗	✗	✓	✓
Online Forums	$	$	✓	✓
Face-to-Face Conferences	$	$	$	✓*
Pre-Conference Workshops	$	$	$	✓*
Event Fee Discounts	✗	20%	20%	20%
Other Event Site License Discounts	✗	✗	20%	20%

**See www.eLearningGuild.com for details*

✓ = Included in Membership ✗ = Not available $ = Separate fee required

The eLearning Guild organizes a variety of important industry events...

THE ELEARNING GUILD 2007 THE ANNUAL GATHERING

April 10 - 13, 2007
BOSTON

LEARNING MANAGEMENT COLLOQUIUM

April 11 & 12, 2007
BOSTON

THE ELEARNING GUILD'S ONLINE FORUMS

CHECK ONLINE
for topics and dates!

DEVLEARN 2007 THE ELEARNING DEVELOPERS' CONFERENCE & EXPO

Fall 2007 Dates TBD
WEST COAST, USA