HTML5 for Ozlab
Camilla Lamberg
ISGC03 Future web standards and mobile multimedia
spring-11 Karlstad University

HTML5 for Ozlab

Table of contents

Vocabulary

TL = short for test leader, the person who has the wizard-role during a test session.

TP = short for test person, the person sitting by the TP-computer during a test session.

Wizard-of-Oz = the technique used for making usability test in Ozlab.

Ozlab setup = the actually set up where tests with Wizard-of-Oz technique is conducted.

Prototype = a graphical, unprogrammed version of a system, made for (usability) testing.

Interaction shells = prototypes used in the Ozlab setup, can also be referred to as Oz-prototypes.

1.Introduction

This paper discusses the possibilities of incorporating HTML5 as the format in which Oz-prototypes are made. Oz-prototypes, or *interaction shells* as they more often are called, are the kind of prototypes used for usability testing in the Ozlab setup. The Ozlab setup is a system developed at Karlstad University which facilitates usability tests according to the Wizard-of-Oz technique. The Wizard-of-Oz technique is a way of making the test person (TP) think that they are testing a fully functionally working product, while all the interaction really are provided by a test leader (TL) sitting in the next room. Details on Wizard-of-Oz technique and Ozlab are found in chapter 2.1, Background of Ozlab. The problem in Ozlab today is that the whole setup is software-dependent to Macromedia Director MX, its file format *.dir* and its scripting language LINGO. This is a severe problem because Macromedia Director MX was bought by Adobe in 2005, and is no longer backward compatible. Macromedia Director is the software in which the interaction shells are currently made and the template file used for the interaction shells is a *.dir* file. Because the amount of people who has knowledge of macromedia Director MX gets smaller and smaller, Ozlab needs to be reprogrammed. It needs to be accommodated to a new software or file format that can work as a long term solution, and requires less prior knowledge about coding and scripting. Unless this happens, Ozlab will become more or less useless in a close future. The goal of this paper is therefore to find out if the HTML5-format could be a good solution for Ozlab. HTML5 is a hypertext markup language mostly used for structuring and presenting contents on webpages, but also works well for offline use. There is also a big number of rapid prototyping tools/software in which you can create prototypes by ”dragging and dropping” graphical components, and then export it to HTML5. This along with the fact that it is backward compatible makes it interesting for Ozlab usage. HTML5 is fully explained in chapter 2.2, Background HTML5.

2. Background

This section covers the background of the Wizard-of-Oz technique, the Ozlab setup, and HTML5.

2.1 Background Ozlab

The Wizard-of-Oz technique is used for faking interactivity in prototypes during user tests. When conducting a user test using the Ozlab set-up the test person (TP) is under the impression that he or she is testing a partial and fully programmed software or homepage, but what really happens is that a test leader (TL) provides for all the interaction:

"In Wizard- of-Oz experiments the test manager sits in the next room interpreting the users' commands and provides the system's responses." (Pettersson 2002)

The idea of Oz-prototypes, and all prototypes in general for that matter, is that they should be built fast and simple without having to waste time or money on a lot of coding when conducting a usability test. It is also important to consider that the test leaders using Ozlab vary from people who were involved in the development of the technique to first year interaction designer students.

"The benefit for designers, whether they are professional or inexperienced, of avoiding programming is the ease with which alternative proposals can be produced for discussions with peers and/or testing with prospective end-users" (Petterson 2002)

Macromedia Director MX is the current software used to create these interaction shells as the Oz-prototypes are called in the present Ozlab setup. Also, both the built-in functionality in the template for the interactions shells and the Ozlab TestRunner itself is based on LINGO, which is the script language used in Macromedia Director MX. This solution is getting outdated because it has changed quite a bit since Adobe bought Macromedia in 2005, and it is no longer backward compatible. The current Director version from Adobe is 11.5 and it cannot be used in the present Ozlab set-up because it is not compatible with the old versions. This means that there are few people left who know how to use Director, and even fewer who starts learning it now. This also means that the interaction shells need to be in the .dir format otherwise it will not work in the Ozlab Testrunner since the Testrunner is programmed to use .dir files only.
Even though Macromedia Director was probably a good option to choose when Ozlab was developed, it is now time to replace it with something else that is more likely to work as a long term solution and minimizing the risk of Ozlab becoming software-dependent again. The new solution should either be based on software that most people are familiar with, and/or software which is really easy to learn without much prior knowledge in programming or web design, and that is backward compatible, or it could be based on a specific format that is more software-independent.

2.2 Background HTML5

HTML5 is the latest, and still developing version of the markup language html (originally created in 1990). It is primary used for structuring and presenting content for the World Wide Web but this paper obviously focuses on offline use of this format. HTML5 is short for HyperText Markup Language version 5 and the basic structure of HTML documents are just about the same in all versions. A basic HTML-document could be written in any text editor such as Microsoft's Notepad so

long as it is saved as an html document. This however requires that the user has a lot of prior knowledge about HTML, and that is something that is not desirable for Ozlab. Also, the whole point of making prototypes and mockups such as the Ozlab interaction shells is to eliminate as much coding as possible to save time and money. There is however a number of different software tools, primarily used for web development or prototype making, that allows you to create GUI mockups that are possible to save or export to html5 documents.

A leading principle in the development of html5 is to make it backward compatible which means that if software such as the Ozlab Testrunner is accommodated to the html5 standard it will also be able to read documents in a previous html version.

As mentioned earlier, the basic structure of an html5 document has not changed much since the first version of html. An HTML-file consists of *elements,* which means different kinds of contents defined by *tags* in the text structure. Each element has an *opening tag* and a *closing tag.* The *opening tag* defines what kind of content is in the element, after comes the content, followed by the *closing tag.* You can think of the tags in an html5-document as boxes, the html-tag is the biggest box and everything inside of it is a smaller box, which also can contain even smaller boxes. For example, if you want to make a basic html5-document with a paragraph/block of text with an image somewhere inside of it like Figure 1 shows, you would write this string of code:

```
<html>
<p>this is the paragraph tag which holds block text and in it is an image tag which
looks like this<img src="nameofimage.gif" alt="name of image" /> the image tag has a
path to the folder where the image is.</p>
</html>
```

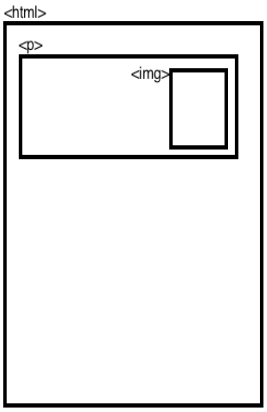

Figure 1. The structure of an html document.

Even if t syntax of html5 is not noticeable different from the previous versions there are however some new elements and attributes added. For example the <nav> tag is new and defines a

navigation link. Also, the <object> tag has been replaced with <audio> and <video> tags so instead of using the <object> tag and then define the datatype you can now use the <audio> or <video> tag right away. If you prefer to use the <object> tag or other old tags, this will also work just fine because version 5 is backward compatible. You can also define articles, command buttons, graphics and last but not least drop down lists with their own tags.

There are more new elements than the ones mentioned above, but I think that these ones are the most important for Ozlab. The fact that it is made easier to add different kinds of multimedia into a document can be an advantage as it makes it easy to make a prototype more trustworthy. Even if the idea is that you should not have to write any code when making an interaction shell it is a good thing that it is very easily done if it is needed.

3. Current system description of Ozlab

This chapter explains how the Ozlab setup works, it describes the different computers and software components in the setup and how they are connected with each other. The template used in Director when building interaction shells is also explained which gives a clear image of what a future interaction shell in html5 needs to include.

Figure 2 shows a setup with a TL- and a TP-computer. To the right a TP sits and watches a screen on which she thinks a fully working program is displayed. This is the TP-computers only purpose, to show the TP the prototype. Everything that happens on the screen is controlled by the TL that sits by the TL-computer in the next room, both these computers has the Ozlab Testrunner installed on them and they are connected to each other by the multiuser server software and the Ozlab file updater which are all installed on the TL-computer. The Testrunner is installed on both the computers and provides the Wizard-of-Oz functionality to the TL-computer.

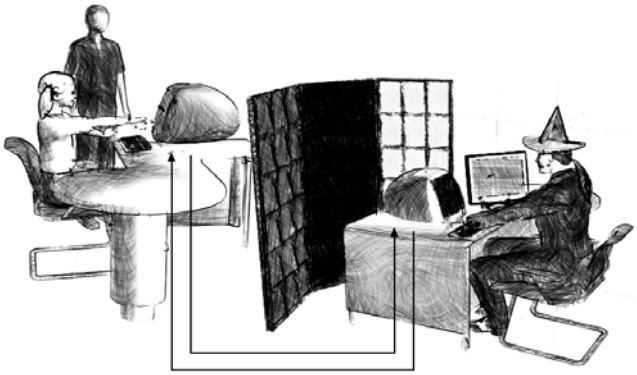

Figure 2. Example of how an Ozlab can be set up. Illustration by Anders Karlsson. © Karlstad University

Figure 3 shows the connection and signal chart within the Ozlab network. In this chart, things such as keyboards, mice, microphones and other external recording devices are not included.

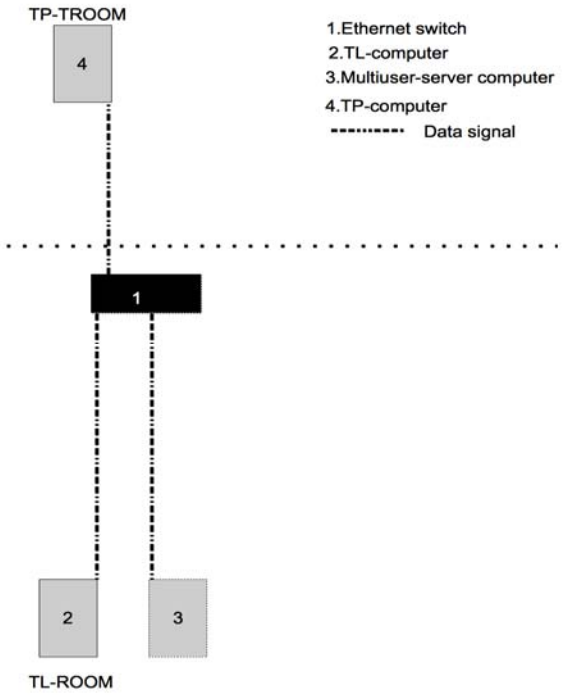

Figure 3. Connection chart.

3.1 The Ozlab setup1.4

The Ozlab Setup 1.4 is where you make all the settings needed for the Ozlab Testrunner to work properly. It is therefor important to check these settings before each new test session. The settings controlled by the Ozlab setup 1.4 are:

- The multiuser Server computers IP-number
- The path to the Ozlab Testrunner root folder on the TL-computer.
- The path to the Ozlab Testrunner root folder on the TP-computer

These settings are saved in the sync.ini files which are explained in the next section.

3.2 Ozlab FileUpdater 1.4.2

The Ozlab FileUpdater is installed on the TL-computer and makes sure that the files in the Ozlab

Testrunner on this computer are exactly the same as the ones on the TP-computer. In order to make a successful test session in Ozlab, the exact same system files and interaction shells need to be available on both the TL- and TP-computers. When running the FileUpdater, two start-up files are created. These files lands up in the root folders of both the TL and TP-computers. The files are saved with the ending .ini but the contents differ between them because they contain the parameters that determine if the computer acts as a TL- or a TP-computer. The Multiuser-Server-computer's IP-number is also saved in these files. The FileUpdater uses the current information in the Ozlab Setup 1.4.

3.3 Ethernet-switch/The Ozlab network

The Ethernet-switch holds the Ozlab network consisting of the TL-computer, TP-computer and multiuser-Server-computer together. This switch is also connected to the University central file system and its internet connection. The Ethernet switch receives and forwards data packets from all of the computers in the Ozlab network along with the university central network.

3.4 Macromedia Director and Ozlab

3.4.1 Creating interaction shells in Ozlab

In order to conduct a Usability test in the Ozlab set-up an Interaction shell needs to be created. These shells are currently made in Macromedia Director MX and are based on a template made especially for the Ozlab set-up. This chapter covers the basics of Macromedia Director MX and the use of it in the Ozlab set-up.

3.4.2 The template and Director

To build an interaction shell that works properly with the Testrunner a certain template .dir file is used as a base. This template has a particular Cast filled with built in function- cast members that are needed in an interaction shell in order to make it work properly with the Testrunner (see chapter 3.5). Below is a list of these castmembers.

They are all scripted with their Swedish names and do not work translated.

objektFlyttbartAvTL Makes an object moveable by the TL
objektFlyttbartAvTP Makes an object moveable by the TP
objektFlyttbartAvTLTP Makes an object moveable by both TL and TP

objektGömbartAvTL Makes it possible for the TL to hide an object from the TP

objektKnapp Creates a clickable button out of any object

objektKomIhågPosition Makes is possible for the TL to bring an object back to its original position by clicking the restore button

objektOsynligtFörTP Makes an object permanently invisibly to the TP

objektOsynligtFörTL Makes an object permanently invisibly to the TL

sidmarkör Is used as pause-function in the interaction-shell. This function is made by adding code in the *framescriptchannel* in the Score-window. This needs to be placed in the same column as a marker in order to work properly.

spelaUppLjudFörTP Allows the TL to control the playback of an audio-file by clicking on diretorobjects of the type button.

textfältEditerbartAvTL allows the TL to edit an object of the type text-field, can be combined with the function below

textfältEditerbartAvTP allows the TP to edit an object of the type text-field, can be combined with the function above.

Saving the Interaction shell.

To avoid overwriting the template it has to be saved under a different name then the one it had before you started creating an interaction shell with it.

3.5 The Ozlab Testrunner

The Ozlab Testrunner is the software that provides the Wizard-of-Oz functionality to the TL-computer, it is also installed on both the TP-computer and runs on both of them at the same time during a test session. The content displayed however looks a bit different. On the TP's computer the graphic elements in the interaction shells are shown and it looks like an interactive program, while on the TL's computer there are a lot more details displayed. In the center of the TL's screen a copy of the TP's screen is displayed. On the rest of the TL's screen the functions the TL needs to conduct the test is placed out in different windows. These windows are explained below. In Figure 4 screen dump, all names are given in English.

The "Byt Skal" window

In the window called "Byt Skal", all the interaction shells that are in the same files are listed. The TL can use this window to change to another interaction shell during the test. The active shell is marked as in the image below. If the TL needs an extra couple of seconds to do some thing without revealing him/herself to the TP there is a button called "vänta" that displays a message to the TP to make him or her think that the program is thinking.

The "Markernavigering(sidnavigering)" window

This window is used by the TL to navigate between the different pages in an interaction shell.In the .dir file these pages are called *markers*. So what really happens in the .dir file when the TL navigates in by using this window is that it jumps from one place on the timeline to another. In this window there are two different ways of navigating, one drop down menu and two buttons, one to go forward and one to go backward.

The "aktivitetslista"

This window shows to TL all the clicks that the TP makes during the test session.
This does however require that the buttons in the interaction shell has been provided with the behavior that registers the clicks in the aktivitetslista.

The "Kontrollpanel"

This window is a control panel with a number of buttons to help the TL moderate the test. Below is an image showing the "kontrollpanel" with all of its buttons, followed by short explanations of each button.

Visa/vänta bild : shows the text "Wait" on the TPs screen
Frys TP skärm: freezes the TPs screen and changes the cursor into an hourglass.
Lås Objekt: locks all the moveable objects in the interaction shell, this lock is marked for the TL to see.
Återställ Objekt: Resets all the objects that have been moved or changed during the test to its original state.
Visa Aktivitetslista: Shows the window "Aktivitetslista" on the TLs screen.
Byt Skal: Shows the window "byt skal" on the TLs screen.
Återställ skal: Restarts the entire interaction shell from the beginning.
Avsluta: shuts down the Ozlab Testrunner on both the TL and TPs computer

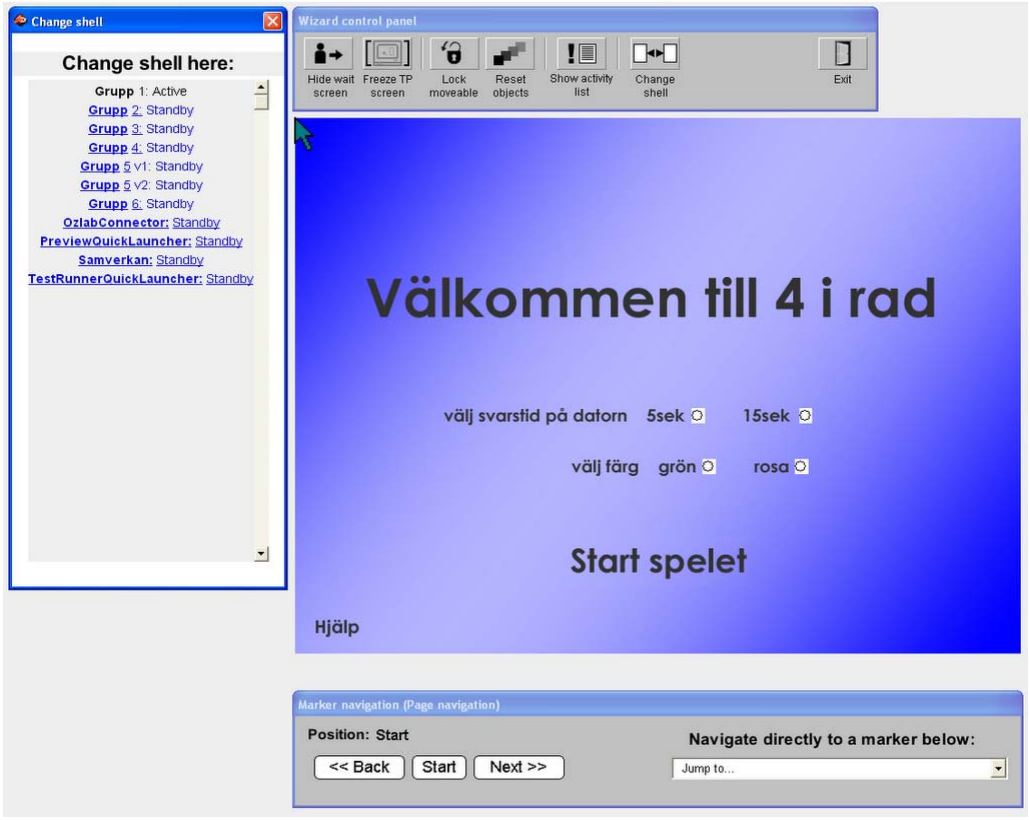

Figure 4. Screenshot of the English version of Testrunner start screen without the activity list.

3.6 The multiuser server computer

Multiuser Server 3.0 is the server-software used in the Ozlab network. It transmits all the communication between the other computers.

4. HTML5 in Ozlab

(1) In order to use html5 documents as interaction shells in Ozlab, the Testrunner needs to be reprogrammed. As mentioned earlier, it is now only capable of reading .dir-files and LINGO-script. To run an html5 document, it would have to behave like a web-browser. Even if the whole point of prototyping is to avoid coding, the Testrunner should also be able to read javascript and php since it is not unusual to add functionality to a prototype to make it more advanced after testing it once, and then test it again in iterations. Thus, to be able to add some functionality would be an advantage.

(2) To make the navigation easier for the TL during a test session, one solution could be to have a template document with a certain number of pages that are already linked together with simple link-tags. You just simply fill these pages with your different content without having to make the connections each time. Let us say that we have a template html5-document with 10 pages in it, then these 10 pages are placed in the same folder on The TL-computer and will therefore be connected even when they are offline. All of these pages hold a default set of link-tags, connecting all the pages to each other.

(3) To elaborate on the 10-page-template idea: let us assume that they will also hold default tags for graphical elements containing the path to the folder in which images, logos or other graphical parts should be saved in order for them to be accessible offline. This would mean that all that needs to be done when creating an Ozlab interaction shell is to open the template in some kind of drag-n-drop prototyping tool, add the content of choice into the pages, and finally remove the pages and links that will not be used (or just keep them for next iteration cycle[1]). After this the document is saved as an html5-document and placed in the right folder on the TL-computer as in (2) while all separate images, sounds or other graphical elements are saved in a separate folder on the TL-computer which the pages in the template holds the path to.

(4) The built in functionality in the current template would also need to be implemented into the new one, and the easiest way of doing this is probably javascript and css. For example, the problem with hiding some objects from the TP that are visible to the TL could be solved by using two different css-sheets, and use javascript to update the TP-computer screen based on actions from the TL. The functionality within the template that is now provided by specific castmembers in Macromedia Director MX should be possible to provide in an html5 template using javascript and css-sheets.

(5) Another thing that needs to be done in order to use HTML5 for Ozlab is choosing a prototyping tool/software to make the interaction shells in. The good thing about HTML5 is that there are several options, both open source and commercial, and more of them keeps popping up all the time. Even if many of these tools have their own file format almost all of them can export to html5 or an earlier version of html. Most of these tools are totally drag-n-drop based, and many of them fake the appearance of different operating systems such as Windows, Mac OS X or Ubuntu. This means that it is really easy to change from one tool to a newer and better one. Possibly, each person creating an interaction shell chooses whichever software fits him/her. Also, there should not be a problem opening an interaction shell made in one tool in another one, so long as it is exported to html-format. Regardless of which software is used, the idea is to use the pre-built template, open it and fill it with the graphical elements made especially for the current prototype and then save the interaction shell

[1] Suggestion from John Sören Pettersson.

with all its components in the pre-determined folder on the TL-computer, in order for it to work offline.

5. Conclusion

Because it is of great importance that the future solution for Ozlab is more stable in a long term perspective it is important not to make it software-dependent. Considering this, HTML5 is a very good option for Ozlab since the actual interaction shells could be built in almost any rapid-prototyping tool, web design tool, or even in a text editor as simple as Notepad. The idea is of course not to build them in Notepad as the whole point of prototyping is to get away from coding, but I just want to point out that it would actually be possible to make a HTML5-based template, with javascript an css-sheets attached in a software as simple as Notepad. The program OzlabTestrunner would however need to be reprogrammed. It would basically need to behave like a browser and be able to read HTML-documents with javascript and css attached to it.

Another thing that makes HTML5 a relatively safe choice for the future is the fact that it is backward compatible and will most likely continue to be. A possible downside with using HTML5 could be that it would be hard to keep updated on which software that is currently the best one to use for building the shells, because there are so many web tools and they are constantly developing. On the other hand, no matter which software you use it will work just fine as long as it is based on the template accommodated to Ozlab Testrunner and everything is placed in the right folders on the TL-computer before a test session.

From my point of view as an Ozlab user, I would really like to try and work with HTML5 in Ozlab. I think it would be a huge advantage, especially for the teachers using Ozlab in courses with first year interaction design students, to be able to tell the students that they can use whichever software they like when they make Oz-prototypes so long as it can open the Ozlab template and export to HTML5. This is much better than starting off with teaching them Director or some other program that will eventually be discontinued. It would also be nice to be able to send a prototype between people in a working team without having to worry about them having the same programs installed on their computers.

I tried to find a really smart quote to end this essay with, but I could not find anything. Thus, here is an empty HTML5 quote tag with absolutely nothing in it.

<q> </q>

6.References

Pettersson J.S. (2001) http://www.is.kau.se/~jsp/ozlab/n.php goto=Swe/Presentation_av_ideerna_bakom_Ozlab_010701.html

Pettersson, J.S. (2002) "Visualising interactive graphics design for testing with users". *Digital Creativity* vol 13 (3): 143-155.

Siponen, J., Pettersson, J.S., Alsbjer, C. (2002). *Ozlab Handhavandemanual.* Karlstad: Institutionen för infomationsteknologi, Informatik/Centrum för HumanIT, Karlstads Universitet.

Siponen, J., Pettersson, J.S., Alsbjer, C. (2002). *Ozlab Systembeskrivning*. Karlstad: Institutionen för inromationstekonologi, Informatik/Centrum för HumanIT, Karlstads Universitet.

W3C http://dev.w3.org/html5/spec/Overview.html

W3Schools http://www.w3schools.com/html5/html5_reference.asp