Vrije Universiteit Brussel

Faculteit Wetenschappen
Departement Informatica
en Toegepaste Informatica

A WYSIWYG Template Authoring Solution for the MindXpres Presentation Tool

Jeroen Heymans

Promotor: Prof. Dr. Beat Signer
Begeleider: Reinout Roels

Augustus 2013

Abstract

Wereldwijd worden er elke dag meer dan 30 miljoen PowerPoint presentaties gemaakt. Met verschillende alternatieven ter beschikking, worden presentatie programma's op vele vlakken in ons leven gebruikt. Ze worden vaak gebruikt in het bedrijfsleven maar ook bij het lesgeven worden ze onmisbaar. Het presenteren van visuele informatie door een presentatie te projecteren die ooit ontworpen was op een computer, is een de facto standaard geworden. De populariteit van deze manier van presenteren groeit nog altijd.

De manier waarop mensen informatie presenteren is grotendeels beïnvloed door de programma's die ze gebruiken. Hoewel de meest populaire presentatie programma's de gebruiker dwingen om informatie in sequentiele slides te presenteren, bestaan er andere methodes om een collectie van informatie te presenteren. Dit wordt aangetoond in het onderzoeksproject MindXpres waarbij de auteurs wegstappen van de traditionele manier van presenteren en gebruik maken van nieuw onderzoek in informatiebeheer en informatie visualisatie. Door kritische evaluatie van het presentatie proces kwamen ze met een lijst van eisen en aanbevelingen voor het ideale presentatie programma.

In MindXpres onderscheidt men drie fases in de tijdspanne van het maken en gebruiken van een presentatie: voor, tijdens en na. Deze thesis focust op de eerste fase: voor de presentatie. Het doel was om de best mogelijke interface te maken voor het MindXpres programma om zo presentaties en themas te maken en te beheren. Om dit doel te bereiken zijn we gestart met een onderzoek en vergelijking van de huidige populaire presentatie programma's. We stelden een lijst van problemen op die aanwezig zijn in deze programma's en baseerden ons ontwerp voor ons eigen presentatie programma hierop.

Om de juiste keuzes te maken bij het ontwerpen van het programma, hebben we ons gebaseerd op vorig onderzoek. Het geïmplementeerde presentatie programma bestaat eigenlijk uit twee grote delen: een presentatie bewerker en een thema bewerker. Deze twee onderdelen zijn aan mekaar gelinkt dankzij vier nieuwe concepten die we introduceren in deze thesis. Huidige presentatie programma's limiteren de gebruikers in het apart beheren van presentaties en themas. In onze oplossing hebben de gebruikers de nodige flexibiliteit in een intuitieve interface. De geïntroduceerde nieuwe concepten kunnen gebruikt worden door huidige presentatie programma's omdat ze een hogere gebruiksvriendelijkheid garanderen.

Naast het ontwerpen van een presentatie programma voor MindXpres hebben we ook een eerste prototype geïmplementeerd. Deze versie is volledig gebouwd in de meest recente Internet technologie and biedt de gebruiker veel functionaliteit aan. Als basis voor het project moesten we een goede JavaScript library kiezen. Om de juiste keuze te maken hebben we een uitgebreide vergelijking tussen verschillende libraries uitgevoerd. We hebben zo kunnen identificeren welke library gebruikt kan worden voor een grote Rich Internet Application. Naast het kiezen van de juiste library hebben we ook een architectuur ontworpen waarvan we overtuigd zijn dat het een goede basis is voor een nieuwe Rich Internet Application waarbij zaken zoals modulariteit centraal staan.

Door weg te stappen van de traditionele manier om presentaties en themas te bewerken, zijn er uitbreidingen mogelijk op verschillende niveau's. We introduceren nieuwe concepten die een meer flexibel beheer toestaan op het gebruikers niveau. De vooropgestelde architectuur staat dan weer betere onderhoudbaarheid toe op het programmeer niveau. Het resultaat is een programma dat zowel een hogere gebruiksvriendelijkheid garandeert voor de gebruikers om MindXpres presentaties te beheren als ook een hogere gebruiksvriendelijkheid voor programmeurs om de functionaliteit van het programma uit te breiden.

Vrije Universiteit Brussel

Faculty of Science
Department of Computer Science
and Applied Computer Science

A WYSIWYG Template Authoring Solution for the MindXpres Presentation Tool

Jeroen Heymans

Promoter: Prof. Dr. Beat Signer
Advisor: Reinout Roels

August 2013

Abstract

With over 30 million PowerPoint presentations created every day and different alternatives available, presentation tools are used in many aspects of our lives. They are very important in a business context but also in an educational context. Presenting visual information by projecting a presentation that has been authored on a computer has almost become a de facto standard and its popularity continues to grow.

The way people present information is largely influenced by the tools they use. Although the most popular tools force the user to make use of a sequential slide paradigm, other methods of presenting a collection of information are possible. This is shown in the research project called MindXpres where the authors stepped away from the traditional slide paradigm and use state-of-the-art research in information management and visualisation. They rethought the entire presentation process and as a result came up with a set of requirements for the ideal presentation tool.

In MindXpres, three phases are identified in the timespan of creating and using a presentation: before the presentation, during the presentation and after a presentation. This thesis focusses on the before the presentation' phase. The goal was to design the best possible user interface for the MindXpres tool in order to create and author presentations and themes. To achieve this, we started with an investigation and comparison of the current popular presentation tools. We identified a list of problems with these tools and based the design of our own presentation tool on the gained knowledge and findings.

In order to make good design choices, we based ourselves on previous studies. The developed presentation tool is in fact split into two main parts: a presentation editor and a theme editor. The editors are nicely linked together via four novel concepts that we introduce in this thesis. Current presentation tools limit the user in authoring presentations and themes separately while

in our solution, they have all the flexibility they want via an intuitive user interface. The novel concepts that we introduced can be reused by current presentation tools as they guarantee a higher usability.

Apart from designing a presentation tool for MindXpres, we also implemented a first prototype version. This version is fully based on new web technologies and offers a lot of functionality. In order to choose the correct JavaScript library to be used as a basis for the project, we performed a JavaScript library comparison. In this comparison, we identified which library can be used for a Rich Internet Application. Alongside choosing the correct library, we designed an architecture which we believe would be ideal for building a Rich Internet Application from scratch that focuses on modularity and low coupling.

By stepping away from the traditional way of authoring presentations and themes, we allow enhancements on multiple levels. We introduce novel concepts that allow more flexible authoring on a user level while the proposed architecture allows for a higher maintainability on a developer level. The result is a tool that guarantees a higher usability which will benefit users to author MindXpres presentations and developers to extend the functionality of the tool.

Acknowledgments

I like to thank my promoter Professor Beat Signer and my advisor Reinout Roels. They were always ready to help with problems, to give advice and to push the project in the right direction.

I also like to thank my friends for helping me to light up my life with good advice, interesting discussions and beautiful memories this year. The gift of such huge amounts of friendship is more than I could ever ask.

Finally I like to thank my family. Although we have been through some rough patches, the future will be very bright. Beauty is in the eye of the beholder.

Contents

Listings

List of Figures

List of Tables

1

Introduction

1.1 Current presentation tools

Currently, Microsoft's PowerPoint[1] is still the most popular presentation tool out there. It is estimated that PowerPoint has a market share of about 95%[2]. This is very visible in the number of new PowerPoint presentations created every day which has been estimated to be 30 million [57]. There are several alternatives on the market like Apple Keynote[3], Apache OpenOffice Impress[4], Prezi[5] and so on. All of them are built by big companies like Microsoft and Apple, or have an open source background like Apache OpenOffice Impress.

1.1.1 Microsoft PowerPoint

PowerPoint is the oldest of the presentation tools mentioned above, the original product proposal [28] dates back to 1984 and was done by Robert Gaskin, an employee of the small start-up Forethought Inc. Back then, it was called *Presenter* but with its first release that occurred in 1987 [29], the name *PowerPoint* was born. The original product was designed to create graphics for overhead transparencies. PowerPoint was ahead of its time with support for different fonts, charts and diagrams. Instead of having to rely on a typewriter operator to create presentations as was standard use in those days, the presenter could now create their own presentations. Originally PowerPoint was created for the Apple Macintosh and as the Macintosh got more advanced, the software grew along by adding the support for colours and other novel functionality.

Microsoft acquired Forethought Inc. in 1987 and the first Windows version of PowerPoint was built. As development was continued, PowerPoint started getting features like real-time projection from a computer, slide transitions and animations. PowerPoint thus started to move away from the initial idea of only creating overhead transparencies. Presentations started to get more advanced features that were only available on a computer screen. Computer technology in general started to get more advanced in the early 90s and Microsoft continued to update PowerPoint with extra features like audio and video support. One of the first major changes happened in 1997 when Microsoft added Visual Basic for Applications (VBA): a macro language that allowed the users of PowerPoint to program new features for the presentation tool and their presentations. With new support of VBA, we saw the rise of dynamic features like a countdown timer.

Now that the tool has reached the age of 26, we can see that it has grown to a very solid presentation editor tool. It is very intuitive to use as people are used to the concept of slides in presentations. PowerPoint currently supports a wide range of multimedia elements and as its market share is about 95%, the least we can say is that PowerPoint

[1] `http://office.microsoft.com/en-us/powerpoint/`

[2] `http://www.businessweek.com/articles/2012-08-30/death-to-powerpoint`

[3] `http://www.apple.com/iwork/keynote/`

[4] `http://www.openoffice.org/product/impress.html`

[5] `http://prezi.com/`

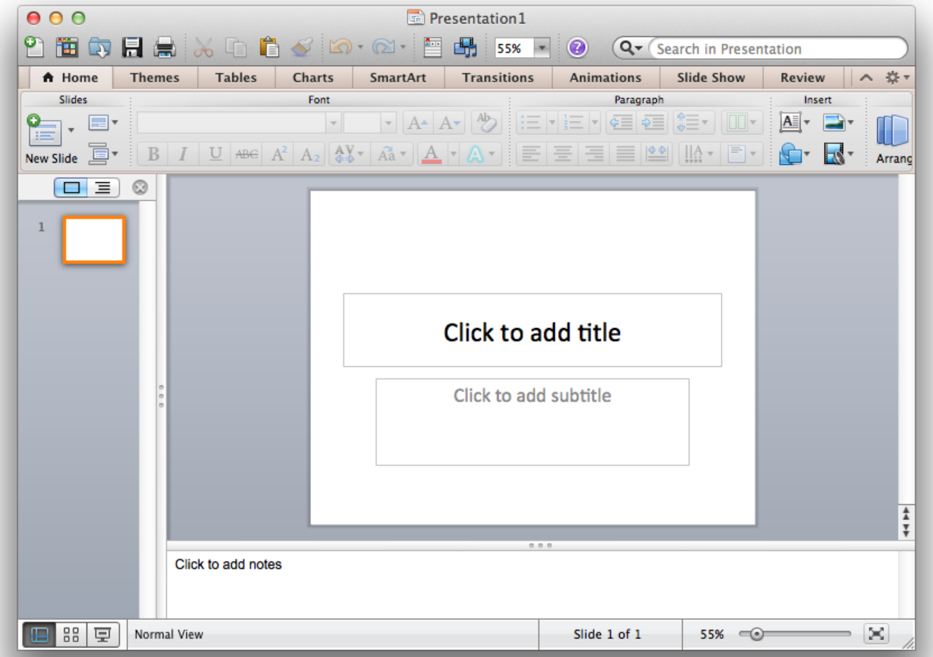


Figure 1.1: The latest version of PowerPoint for Mac OS X

is popular. However, over the years it received a lot of critique that ranges from general critique about the concept of slides [24, 76, 71] to the way PowerPoint has negatively influenced the way people teach and learn [42, 3, 65].

1.1.2 Apache OpenOffice Impress

Another presentation tool that is growing in popularity is OpenOffice Impress. It is part of the Apache OpenOffice office suite that was developed by Sun Microsystems. In the past, many have made a comparison between PowerPoint and Impress for being very similar. Impress is a free open source presentation editor program that is compatible with PowerPoint as it can edit PowerPoint file formats like `.ppt` and `.pptx`. Most people know Impress because it is often provided as part of a default office suite on Linux systems. On the first of January 2013, the OpenOffice suite passed the line of 31 million downloads in total and has now at the end of May passed 50 million downloads[1]. We can conclude that the popularity of this PowerPoint alternative is certainly booming.

[1]`http://www.openoffice.org/stats/downloads.html`

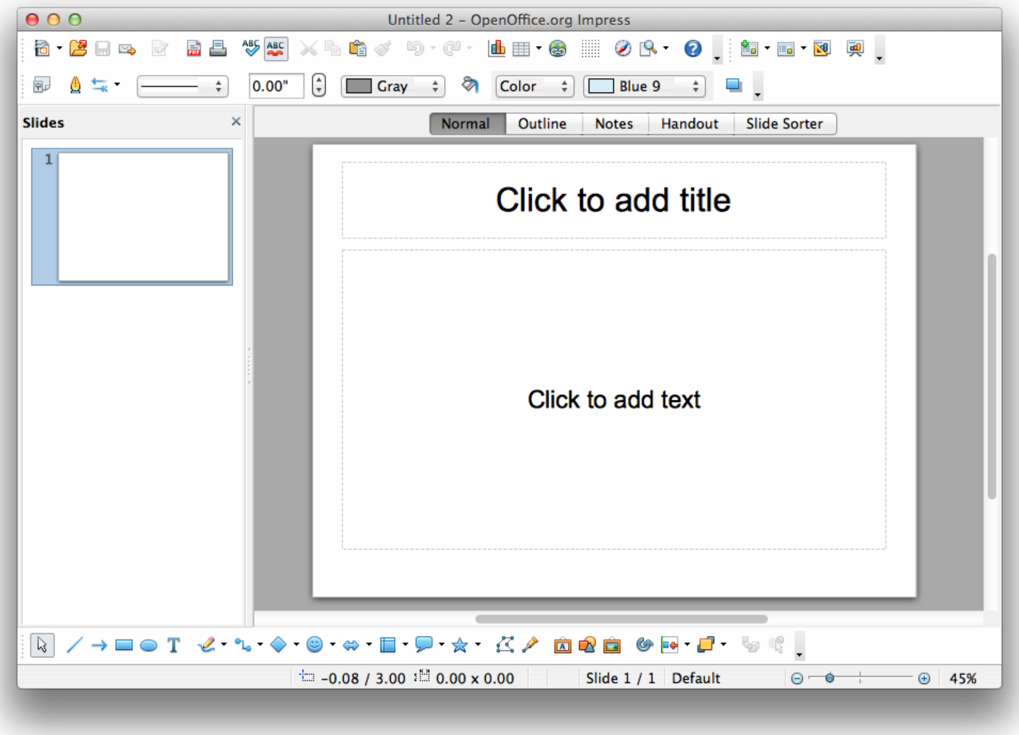


Figure 1.2: The latest version of Impress for Mac OS X

1.1.3 Apple Keynote

Keynote was unveiled on January 7, 2003[1]. At its launch, it had features like import and export to PowerPoint, export to PDF, professionally designed themes, animated charts and tables and much more. This made it a very good presentation editing tool for slick presentations. Since 2005, it is part of the iWork office suite developed by Apple[2]. Apple focused on the more visual aspect of presentations, it wanted the user to be able to create very polished looking presentations. They have done this by focusing more on the typography, good looking themes, smooth animations and so on. In recent years, Apple created a Keynote version of iOS to provide synchronization of presentations between iOS and OS X devices and allows the user to present Keynote presentations with their iOS device.

1.1.4 Prezi

A new kid in town is Prezi. The word prezi is the Hungarian short form of the word presentation. The original version only launched in April 2009, making it one of the

[1]`http://www.apple.com/pr/library/2003/01/07Apple-Unveils-Keynote.html`

[2]`https://www.apple.com/iwork/`

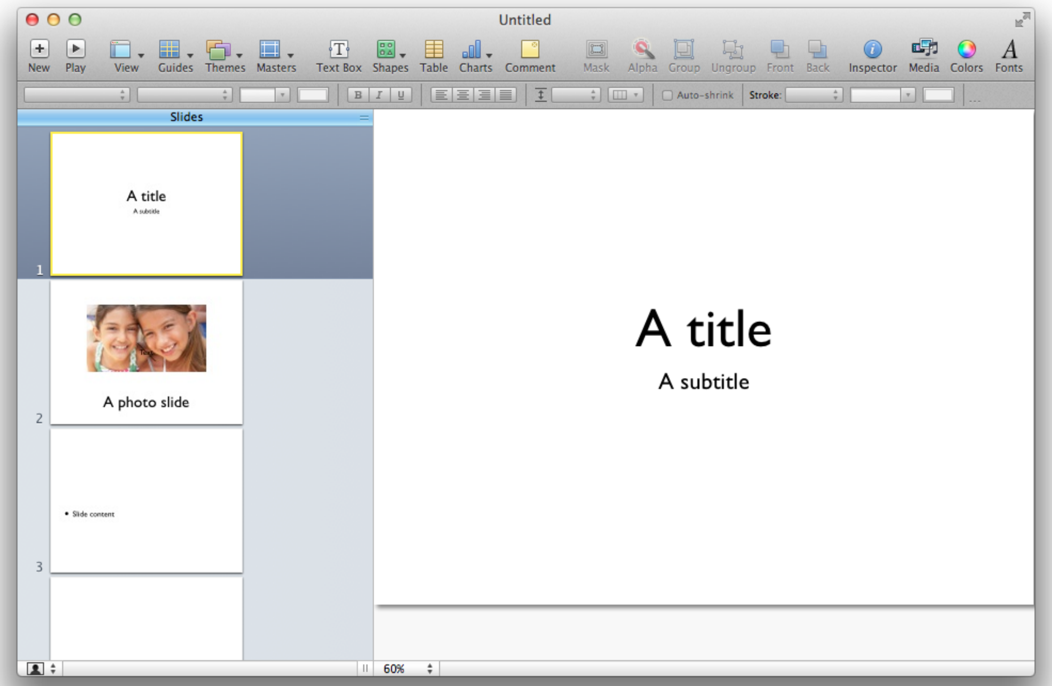


Figure 1.3: The latest version of Keynote for Mac OS X

newest presentation tools available. It is a product of the company called Prezi and offers cloud-based (SaaS) presentation software where presentations are created on a virtual canvas. Prezi steps away from the slide paradigm and provides the user an infinitely large canvas to work on. The idea behind Prezi is that users insert all their presentation elements, like text and video's, and link everything together by defining a path. Important concepts can be represented by using large objects in contrast to details and less important content which may be represented by small objects.

Prezi is one of the few presentation tools that offers a zoomable user interface which allows the users to zoom in and out on their presentations and media elements. This is done in a simulated 3D environment: a 2.5D environment where 2D graphical projections are used to create a simulated appearance of being 3D. In recent versions, a desktop version has been released. In the early days, Prezi was a website-based tool built in Adobe Flash. Thus, users had to use the tool via their browser which needed the Flash plugin. Now with the desktop version, users are able to create and save their presentations offline. As the desktop version uses Adobe AIR, the application is an exact replica of the online version.

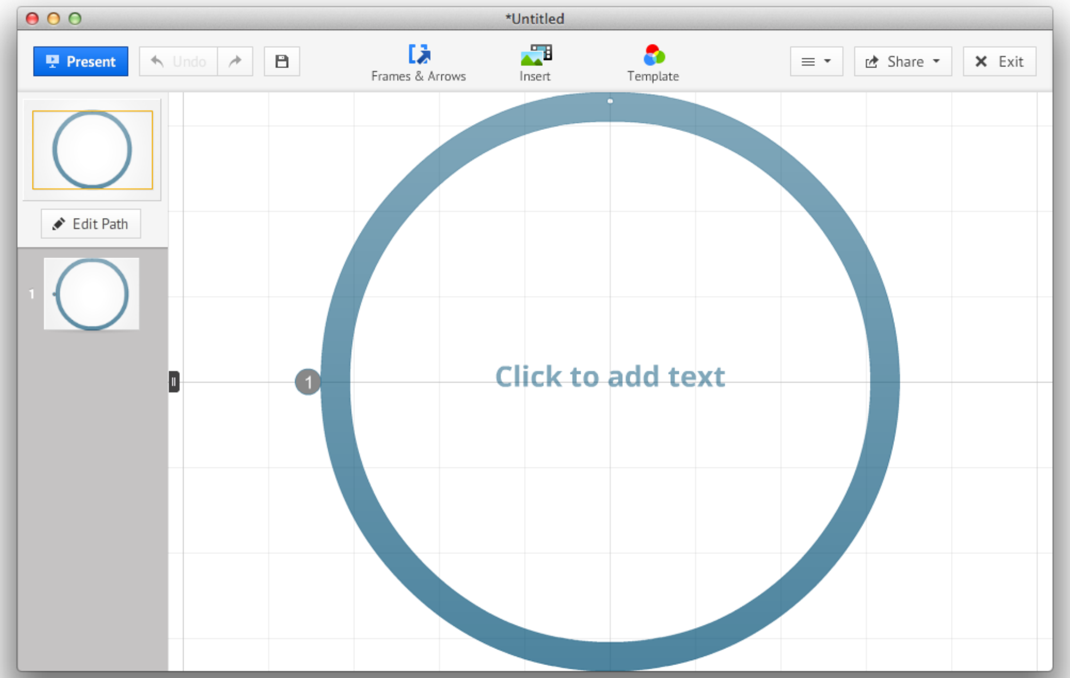


Figure 1.4: The latest version of Prezi Desktop for Mac OS X

1.1.5 Other tools

Prezi is not the first tool to step away from the slide paradigm. Academic work like Fly [45] have a planar interface for authoring presentations that allows authors to lay out information in a map-like fashion on a plane. There also have been attempts at improving PowerPoint in research. Concepts such as those presented in Counterpoint [35, 34] have been introduced to allow PowerPoint to be used with a zoomable user interface.

Another tool with interesting features is Google Docs [1]. It did not move away from the slide paradigm but it offers other notable features such as collaboration. Via collaboration functionality, it allows multiple users to edit documents at the same time. Another interesting thing about Google Docs, is that it works entirely online via the use of HTML5, CSS3 and JavaScript. Unlike with Prezi, which is also a web-based tool, users are not obligated to have plugins like Flash installed.

[1] `https://drive.google.com/`

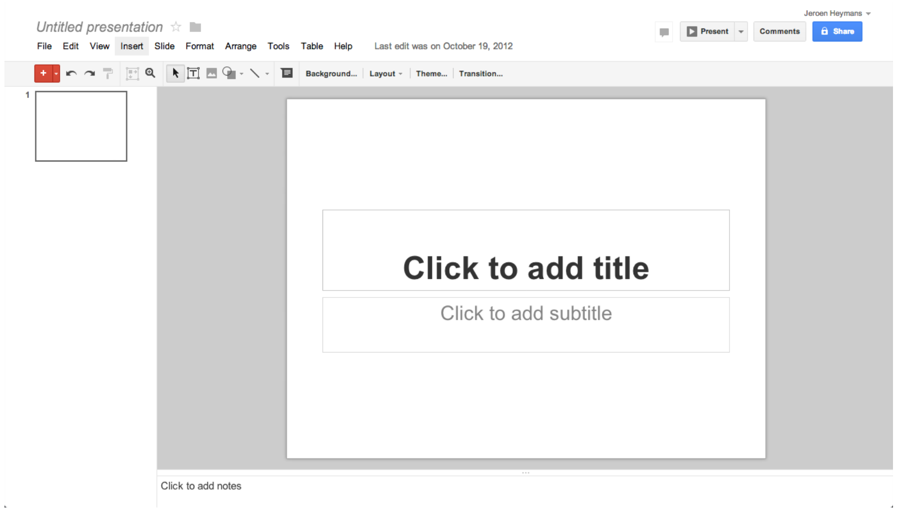

Jeroen Heymans
Untitled presentation
File Edit View Insert Slide Format Arrange Tools Table Help
Last edit was on October 19, 2012
Present
Comments
Share
Background... Layout Theme... Transition...
Click to add title
Click to add subtitle
Click to add notes

Figure 1.5: The interface of Google Docs when editing a presentation

1.2 MindXpres

MindXpres is an extensible cross-media presentation tool [62] currently under development. It addresses many of the issues of current slideware. MindXpres steps away from the traditional slide paradigm and is based on state-of-the-art research in information management, hypermedia and zoomable user interfaces.

The term hypermedia is used as a logical extension of the term hypertext which was first coined by Ted Nelson [54]. Elements like graphics, video, text, audio and hyperlinks are interweaved in a non-linear way. As this is closer to how humans think and learn, MindXpres uses hypermedia instead of unidirectional hyperlinks as seen on the web, to connect elements together. The Resource-Selector-Link model (RSL) [64] has been chosen as the basis for the modelling of hypermedia in MindXpres to allow semantic links, navigational links, structural links, transclusion, annotation and context awareness.

Zoomable user interfaces (ZUI) [8] are graphical user interfaces where the user can change the scale of the view so they see more details of the presented information by zooming in or less details by zooming out. One of the earliest attempts at creating a ZUI as a computer interface was Pad [59] which, after further research, resulted in Pad++ [10, 6]. Zoomable user interfaces are becoming more used every day as popular applications like Google Maps[1], Google Earth[2] and Prezi are getting more mainstream. As MindXpres focuses on providing a canvas for the user to lay out their information, a zoomable user interface is used to provide the best possible control over presentations. The user can either follow a pre-defined path through their presentation or they can navigate via a zoomable user interface in their presentation to jump from element to element.

[1]`http://maps.google.com`
[2]`http://www.google.com/earth/index.html`

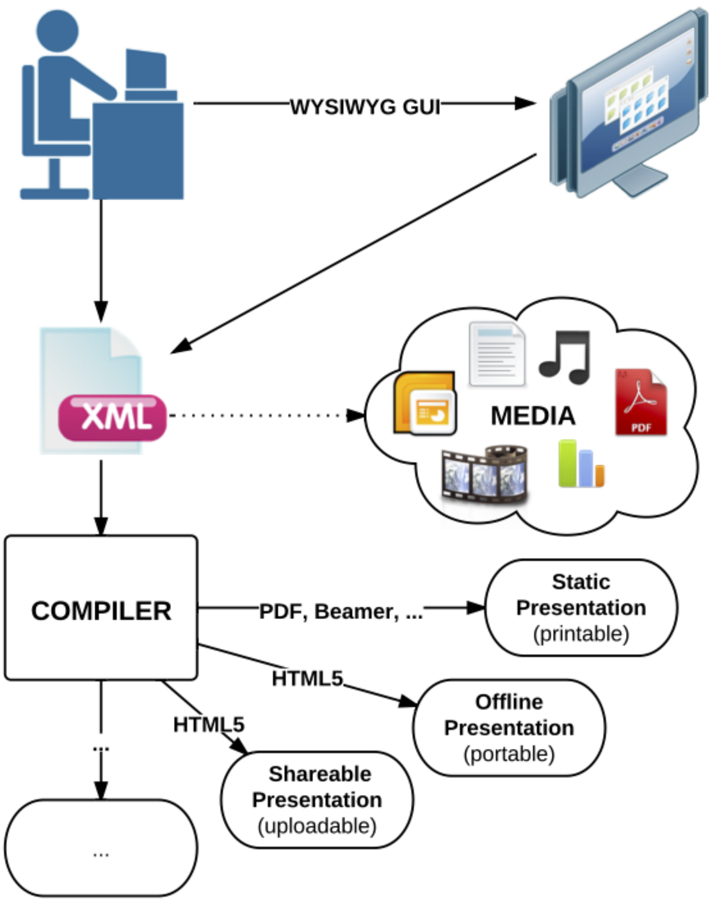

WYSIWYG GUI
XML
MEDIA
PDF
COMPILER
PDF, Beamer, ...
Static Presentation (printable)
HTML5
Offline Presentation (portable)
HTML5
Shareable Presentation (uploadable)
...
...

Figure 1.6: The architecture of MindXpres

The architecture of MindXpres is visualised in Figure 1.6. The user is able to create their own presentations in XML which can be compiled to different output sources like a PDF, an HTML document and so on. Not everyone who creates a presentation wants to manually write an XML file so it should also be possible to use a graphical user interface which generates the XML file for the user. This is the focus for this thesis: the creation of a good graphical user interface for MindXpres. To do this, we looked at how current presentation tools (as introduced in Section 1.1) tackled the implementation of certain features.

2 Problem statement

In Chapter 1 we introduced the currently popular presentation tools. Most of them force the user into the sequential slide paradigm. We strongly believe that this enforced paradigm is a limitation to the flexibility of the user. Thus, we performed a comparison on the current presentation tools in Section 2.1 to identify the limitations of the sequential slide paradigm. An important feature of presentation tools is the possibility to have themes in a presentation. But how are the concepts of themes and presentations linked together? Is it flexible enough or are certain limitations present? If so, how can we solve these limitations?

The web continues to evolve and the concept of Rich Internet Applications is getting more attention. Hence we found it interesting enough to look whether current web technologies are mature enough to use for a large application like a presentation and theme authoring tool. We introduce the current technologies in Section 2.2 in which we try to find answer to the question ‘What are currently the best technologies to use for a RIA?’. User interface languages are used to describe a user interface and possibly are a major part of our implementation. They allow the developers to model, design and implement a user interface. Therefore, we investigate in Section 2.3 the latest user interface languages and their potential for this project.

2.1 Identifying the shortcomings of current tools

The problem with current presentation tools is that we never question them. Many people only know Microsoft PowerPoint as it is the primary tool used when educating people. For many people, the word *PowerPoint* is synonymous for *presentation*, just like *Internet Explorer* was once the synonym for *Internet* or *Windows* for *operating system*.

In order to design a good presentation tool, we took a look at what the shortcomings of current popular tools are. What are mistakes that we can avoid? Which features of current tools can we reuse and improve? We had to avoid that we simply made a presentation and theme editor for MindXpres only based on how we use current tools. For instance, certain areas like the editing of themes could perhaps be improved.

We tested several presentation tools, to be more specific:

- *Microsoft PowerPoint Mac 2011*: version 14.0 build 100825
- *Apache OpenOffice Impress*: version 3.4.1 build 9593
- *Apple Keynote*: version 5.3 build 1170
- *Prezi*: online version[1]

We compared these tools on several points which we classified into two categories: presentation editing and theme editing. The reason behind this is that the goal of MindXpres is to separate the content and presentation. It should be possible to edit, for example, a theme of a presentation separately. We looked at the presentation tools to see if they have such division and how they implemented the presentation and theme editing. Questions we asked ourselves for when the user is editing a presentation are:

- How can one add presentation elements?
- How can one move presentation elements?
- How can one resize presentation elements?
- How can one reuse slides of presentations?
- How can one edit the structure of a presentation?
- How can one choose a different theme for a presentation?
- How can one zoom in a presentation?

The questions that we asked ourselves when the user is editing a theme are:

- How can one edit the current theme of a presentation?
- What are the effects of presentations that use a theme that was edited?
- How can one edit the styling of text elements?
- How can one change the backgrounds of slides?
- How can one get a preview of the theme you are editing?

[1]`http://prezi.com`

2.1.1 Presentation editing

2.1.1.1 Adding new elements

When creating a presentation, one is constantly adding elements: text, images, videos, links and so on. A presentation can be seen as a collection of different media elements that are put in a certain structure. We compare PowerPoint, Impress, Keynote and Prezi to see how they allow users to add new elements to a presentation. When identifying the different input methods, we also evaluate them to see if there are certain shortcomings.

Before we take a look at *how* elements can be added in the different presentation tools, we also look at *what* elements can be added. We have done a basic comparison that is viewable in Table 2.1. It only takes one glance at the table to see that PowerPoint is the clear winner here as it supports the most possible elements. However, we have to make a remark on the results. Some elements belong to slides (like the date/time, slide number and so on) which is obviously lacking in a tool like Prezi as this is not meant to have slides in a presentation.

	PowerPoint	Impress	Keynote	Prezi
New slide	+	+	+	- [1]
Duplicate slide	+	+	+	- [1]
Import slide	+	+	-	- [1]
Section	+	-	-	-
Comment	+	+	+	-
Text box	+	-	+	+
Header	+	-	-	-
Footer	+	-	-	-
Date/time	+	+	-	-
Slide number	+	-	-	- [1]
Picture	+	+	+	+
Audio	+	+	+	+/- [3]
Movie	+	+	+	+/- [2]
Symbol	+	+	+	+
Shape	+	+	+	+
Hyperlink	+/- [4]	+		-
Table	+	+	+	-
Chart	+	+	+	+

[1] Prezi has no notion of slides
[2] Only YouTube video's are supported
[3] Only background music is possible
[4] A hyperlink can only be added to an existing element

Table 2.1: Comparison of the elements users can add in the presentation tools

PowerPoint Adding elements in PowerPoint can be done in three ways: via a menu at the top of the application, via the ribbon or by slots in a presentation itself.

The menu at the top is a very basic method and can be seen in Figure 2.1. It is straightforward and Microsoft tried to make it as easy as possible, that's why they divided the menu in different sections to group elements together. Some of the menu elements are grouped in a very logical way, for example the three possibilities to add slides (new, duplicate and import) are grouped together. Others are not that logically grouped like for example *Text Box* and *Slide Number*. A total of 21 elements in a menu is certainly a lot and some elements in the menu are very abstract. For example, what makes the difference between a symbol and a shape? Certain shapes like arrows also appear in the symbols list. As the organisation is not that clear, this may cause troubles for the users. More advanced users might prefer the elements to be alphabetically ordered while more novice users might prefer a hierarchical structure [50].

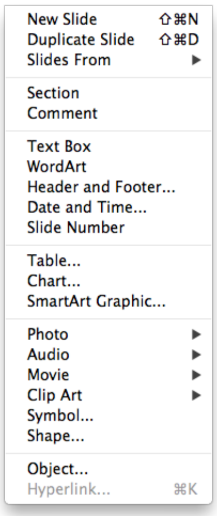


Figure 2.1: The PowerPoint menu that guides the user in adding elements to a presentation

The second way in PowerPoint to add elements is via the ribbon as shown in Figure 2.2. This looks more limited as they only provide four links: `Text`, `Picture`, `Shape` and `Media`. Note that these four links group different elements together. For example via the `Media` link, it is possible to add a video or audio clip. However, this is not that logical as pictures have to be added via the `Picture` link although pictures are also a type of media. The other problem with the grouping in the ribbon is the fact that not all options are available to the user. This is contrary to the menu at the top of the application where all options are available. For example slides and comments cannot be added via the ribbon. In situations where we want to build a presentation based on slides of other presentations, the ribbon is useless.

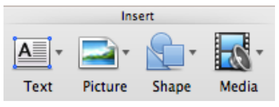


Figure 2.2: Via the ribbon, PowerPoint offers a shortcut to add common elements

The third and final way to add elements in a PowerPoint presentation is via the slides themselves as shown in Figure 2.3. When a user has added a slide, this slide has slot-like elements. In each slot, there are six links which allow a user to add: tables, graphs, SmartArt graphics, pictures, clip art and videos. Again we see a lack of all the possible elements. For example, it is not possible to add audio elements.

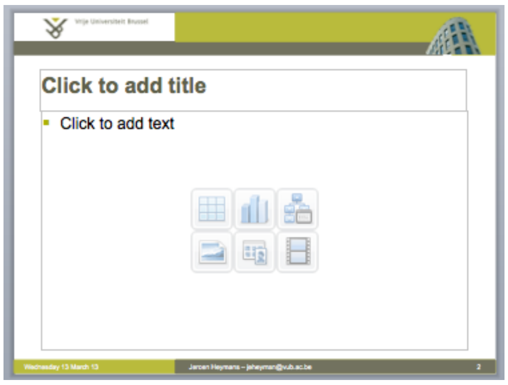


Figure 2.3: PowerPoint shows symbols where you can click on the add content to slides

We can conclude that for adding elements in a PowerPoint presentation, the user is best to rely on the menu at the top of the application as their choice is limited when using any of the two other methods. Combining interface types like the WIMP and the Ribbon can possibly lead to confusion as both types do not support the functionality to add the same elements to a presentation, a conclusion that is also described in [20].

Impress OpenOffice.org Impress offers three ways to add elements to a presentation: via a menu, via a dialogue that can also be turned into a toolbar and via links in the slots on slides.

The first way of adding elements via a menu in the top of the application is just like it is done in PowerPoint and is shown in Figure 2.4. The difference with PowerPoint is that there is less grouping in the menu elements. There is also a difference in what elements one can add (see Table 2.1). For example, it is not possible to import slides from other presentations. We can conclude that the amount of available functionality is different. There is also a grouping in the menu but it is difficult to find out on which basis the grouping took place. A wild guess is that they tried to divide it into *media* and *non-media* elements.

An alternative is to use a small dialog which also can be attached as a toolbar to the top of the application, see Figure 2.5. Here we see the same problem as with the Ribbon interface used in PowerPoint that certain elements can not be added to the presentation via this interface type.

A third and final way to add elements to presentations is just like it is possible in PowerPoint: directly on the slides as can be seen in Figure 2.6. We see here the same problem as in PowerPoint: the choices are limited and there are less possibilities than via the other interfaces.

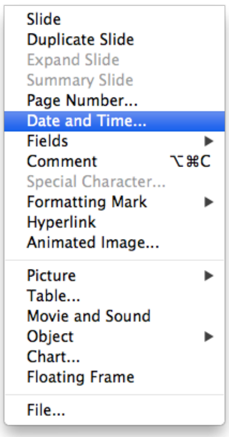


Figure 2.4: A menu at the top of the Impress application allows the adding of elements

Figure 2.5: Via a small dialog that can also be added as a toolbar, Impress offers an alternative to the menu to insert elements

We can conclude that Impress has copied a lot of functionality from PowerPoint. This similarity was even more apparent in the past when the Ribbon interface was not yet introduced in PowerPoint. In those days, Impress could be thought of as a “PowerPoint light”.

Keynote As shown in Figure 2.7, there are multiple buttons available at the top of the application to add elements. They are however very limited and for example, adding media elements is not possible via these buttons. So how can the user add images, videos, audio and so on? To add these types of media, the user has to go via the menu `Insert` and strangely enough has to select the option `Choose...` as we can see in Figure 2.8. This is quite cumbersome and not user-friendly. However, there is a button at the top of the application which allows the user to add media. A problem with this button is that Keynote expects the user to have added their pictures in iPhoto, their music in iTunes and so on. Another possibility to add media to a Keynote presentation is by dragging the original media files from the Finder and drop them into Keynote. Although this allows extra flexibility, the user never gets any hints that this functionality is available. We therefore conclude that although there are multiple options to add media to a Keynote presentation, the possibilities are either limited or rather hidden.

Prezi Prezi is the only one out of the four presentation tools that offers one interface to add elements to a presentation. This is done via a menu at the top as we can see in Figure 2.9. When the user clicks on an element in the menu, a possible dialogue appears for defining the settings. For example, when adding an image, the image itself

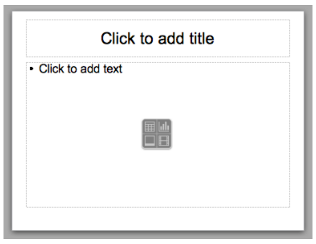


Figure 2.6: There are buttons on the slides available to add elements

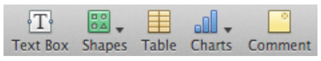


Figure 2.7: At the top of the application, multiple buttons to add elements are available

is requested either via a Google Image search[1] or via an upload. After this, the element is simply added on top of the presentation, somewhere on the screen where the user is editing. A *Processing...* message is shown after which the element is actually shown in edit mode like in Figure 2.10. This is confusing as it introduces a problem: the interface elements in edit mode to actually edit the element are shown over the actual element. In case of adding an image, the user cannot see the actual image itself unless they first exit the edit-mode by clicking outside of the area covered by the element. There is also a notable difference between the different types of elements that a Prezi presentation can handle, as illustrated in Table 2.1.

The menu does not offer that much options and a strange thing is that there is no option to add text. In order to add text, the user has to double click in an empty part of the presentation after which the option appears to directly add text as shown in Figure 2.11.

Conclusion Both PowerPoint and Impress offer three different ways to add elements to a presentation, Keynote offers two and Prezi offers only one. The three options in PowerPoint and Impress are not alternatives for each other as not all input interfaces offer the same options. This makes it possibly more confusing to the users of these presentation tools to know how they can add elements. Even if they know that there are multiple possible methods to add elements it is not clear which method will allow them to add the type of element that they want to add. In Keynote it is also confusing as the user has to go through menus that at first do not suggest that they will allow them to add an image. Adding media can happen via two interfaces: either via a button that forces to have the media already imported in iPhoto or via drag-and-drop functionality that is not really visible to the user. As Prezi only provides one menu, it represents the clear winner as they should not fail in providing a consistent interface. However, adding text is not possible via the same menu.

[1] `http://images.google.com/`

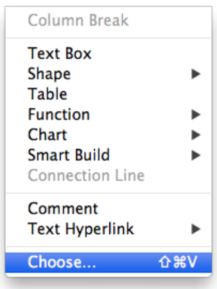


Figure 2.8: There is a menu available in Keynote to add all possible elements

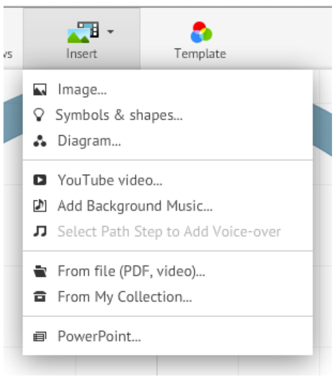


Figure 2.9: Adding elements in Prezi is done via one menu

2.1.1.2 Moving elements in a presentation

When a user has different elements in a presentation, they might want to change the order or the placement of these elements. The reason behind this is simple: presentations evolve and it can not be expected that users will add all elements in their presentation in the right order. We compare the three presentation tools to see how they handle the movement of presentation elements. One interesting aspect to compare in the case of PowerPoint, Impress and Keynote is whether or not they allow the user to move elements from slide to slide.

PowerPoint When hovering with the cursor over an element in PowerPoint, the cursor changes in four small arrows pointing each in a different direction (N, E, S, W). This is the default cursor to show the user that they can move the element they are currently hovering with his cursor. If the user then clicks and holds the mouse button, they can drag the element to a new place. We notice here that this is not always the case. For instance, when the user hovers over a text box, the cursor only changes into a *move-cursor* when the user hovers over the border of the element. When hovering over the actual content of the text box, the cursor changes into a vertical bar to denote that the content can be edited directly. This might be confusing as it is not always immediately clear to the user what the border of a text box is. Some text boxes may not have a border or have a transparent border. Another problem that we have identified, is that it is not

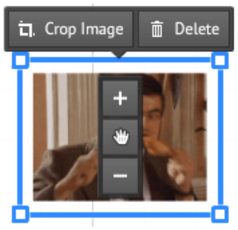


Figure 2.10: The edit mode in which an element can be. Via hovering interface elements, actions can be performed on the element

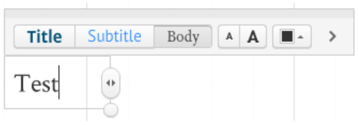


Figure 2.11: Adding text in Prezi can be done by a double click in an empty part of the presentation

possible to move elements from slide to slide. Elements are added per slide and have to remain on that slide. The only way we could find to move an element to another slide without re-adding it, was to cut and paste it. This is not user friendly as it gives certain limitations to the user.

Impress There are two big differences between Impress and PowerPoint. The first one is the fact that the user is not informed that an element is moveable when hovering over it. The cursor never changes and thus the user should assume that he is hovering over an element that is moveable. The second difference can be found when actually moving the element. While in PowerPoint one has to have the cursor changed into the four pointing arrows, the user in Impress can simply click on any element and drag it around. However when using a text box, things are even more different. Like in PowerPoint, the text box can be moved when clicking on the border. The text box is however also moveable when clicking on the actual content. If the user clicks between the text and the border, the element is not moved along with the cursor of the user. Thus we note that there is a sort of “dead space” in text boxes.

Keynote Keynote has a better usability than Impress and PowerPoint when it comes to moving elements. It does not matter where you click on an element, it will always be moved. There is no dead space like in Impress and the moving of text boxes is not limited to the borders like in PowerPoint. Just like in Impress, the user is not informed when they are hovering over an element that can be moved. Moving elements from one slide to another slide is also not possible, just like in PowerPoint and Impress.

Prezi As Prezi does not have the concept of slides, there are only two possible comparison points. How does the application allow the user to select elements to move them and how does the user get informed that they are hovering over a moveable element? The user gets informed by a minimum bounding box that appears when hovering over elements. Via this, the user immediately knows that they can move the element and also where they can click to move the element. When they click in this minimum bounding box and starts dragging the cursor around, the element itself is moved.

Conclusion The clear winner here is Prezi as it solved all the issues that appeared in PowerPoint, Impress and Keynote. Elements get a minimum bounding box around them when the user hovers over them, thus notifying the user where they can click to move the element. Clicking on the element is also not limited like it is the case in Impress where there is a dead zone or like in PowerPoint where one is limited to the border of a text box.

2.1.1.3 Resizing elements in a presentation

One of the first things most users do when they have added a new element to a presentation, is changing the size of that element. The reason is simple: elements have a default size and may not fit in correctly at first. We compare the four presentation tools to see how they tackle the resizing of presentation elements and what the possible shortcomings are.

PowerPoint Resizing elements in PowerPoint can be done after selecting them. This shows extra interface elements which can be seen in Figure 2.12. Everything acts like one would expect: when clicking on one of the dots in a corner, the element is resized in width and height at the same time. Like in many applications, when holding the *shift* key, the aspect ratio will be preserved. This is mostly a feature that users want when resizing an image. The strange thing is that in PowerPoint, the aspect ratio is preserved by default when resizing an image based on one of the dots at the corners of the image. The user is thus limited when resizing images. It is for instance not possible to make an image two times as wide and half as high in one movement.

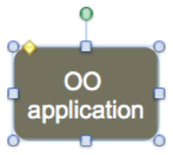


Figure 2.12: Resizing elements in PowerPoint is possible via the small dots after selecting the element.

Impress To resize elements in Impress, they first have to be selected. After that, eight small squares appear on the sides to allow the resizing in all directions. Just like one can expect, selecting one of the squares at the corner of an element allows the user to resize in both the height and the width dimension of the element. When holding the *shift* key, the aspect ratio is saved during the resizing. This is especially helpful when resizing images to prevent possible distortion of the actual picture.

Keynote Keynote allows the user to resize elements in the exact same way as Impress. Image resizing however is limited as is the case in PowerPoint: the aspect ratio is always preserved and so it is, for example, not possible to double the width of an image while holding an equal height.

Prezi The resizing of elements in Prezi is quite odd: the aspect ratio is always preserved. Users do not have control over the fact if the aspect ratio should be respected or not. Because of this limitation, only four possible points are available which the user can select to resize an element: one in each corner.

Conclusion There are several notable differences in how the presentation tools have tackled the problem of allowing the user to resize presentation elements. PowerPoint and Keynote preserve the aspect ratio by default when resizing images while Prezi does this with all types of presentation elements. Keynote and Impress have implemented the same features for presentation elements that are not images. These are more flexible: resizing via eight dots to allow resizing in all possible directions and the optional preserving of the aspect ratio by holding the *shift* key.

2.1.1.4 Reusing slides of a different presentation

When a user has to create a lot of presentations that are very similar, they would like to be able to reuse certain slides. As we can see in Table 2.1, not all presentation tools support the importing of slides. A comparison is still interesting to do as the importing itself may be very different for each presentation tool.

PowerPoint In PowerPoint, the `Insert` menu provides the functionality to import slides as can shown in Figure 2.1. When clicking through, a dialogue appears (after the user has selected another PowerPoint presentation) in which they can now select the slides that they want to import. This dialogue is shown in Figure 2.13. Although the user gets an overview of all the slides they can select, the view can not be changed. It is for example not possible to see in more detail what is on the slides. If a user knows their slides very well, this is not really an issue. But if they import the slides of a presentation that they do not know that well, they might get stuck. A welcome feature of the dialogue, is that it does not disappear right after the user has selected slides and clicked on `Insert`. The dialogue remains visible, allowing the user to select, for example, slide 5, insert it and then select slide 3. The user has full control over the sequence of inserting slides.

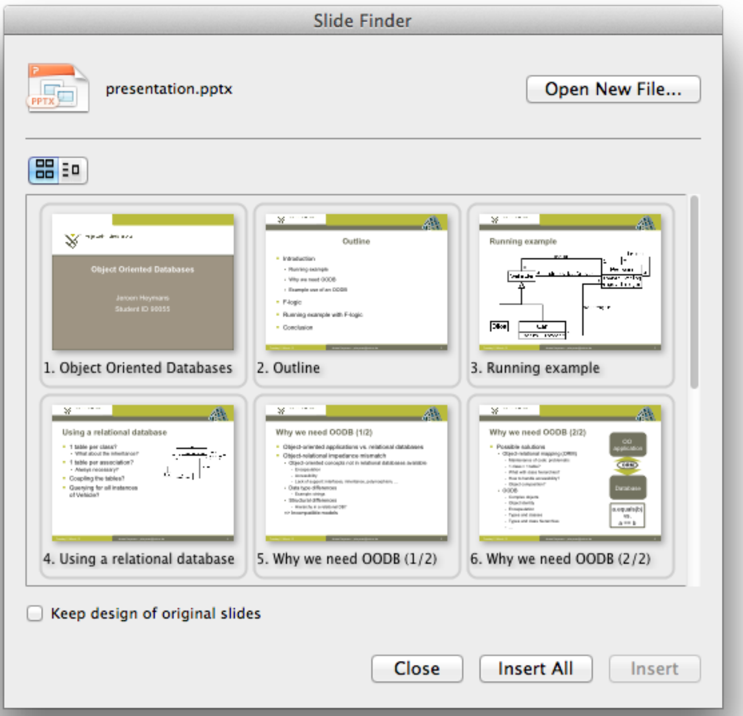

Slide Finder
presentation.pptx
Open New File...
1. Object Oriented Databases
2. Outline
3. Running example
4. Using a relational database
5. Why we need OODB (1/2)
6. Why we need OODB (2/2)
Keep design of original slides
Close
Insert All
Insert

Figure 2.13: Importing slides in PowerPoint works via a selection-menu

Impress Impress allows the importing of slides but this is not that obvious at first. Their menu does not have something like a *Import slides* option. It is however possible via the *File* option in the *Insert* menu. This will result in a dialogue that allows the user to select a file after which they get a dialogue as can be seen in Figure 2.14. We can clearly identify that this is not user friendly at all as only the slide names are shown. There is no preview and in the case the creator of the presentation has not defined a clear slide title, the default *Slide x* is shown. So although the feature is available in Impress, two major problems can be spotted: it is not easily accessible and it is not user friendly.

Keynote Keynote allows the user to import slides in a presentation if the originating presentation is also opened in Keynote. The user is able to simply drag and drop slides between multiple presentations. There is no wizard available to the user and this functionality relies on the knowledge of the user as there is no visual hint that this is possible.

Prezi As Prezi does not support the concept of slides, it appears to be logical to say that the feature is not available. But as slides are parts of presentations, we checked whether it is possible to insert parts of another Prezi presentation or maybe import a presentation that was created in another tool. As we can see in the *Insert* menu in Prezi

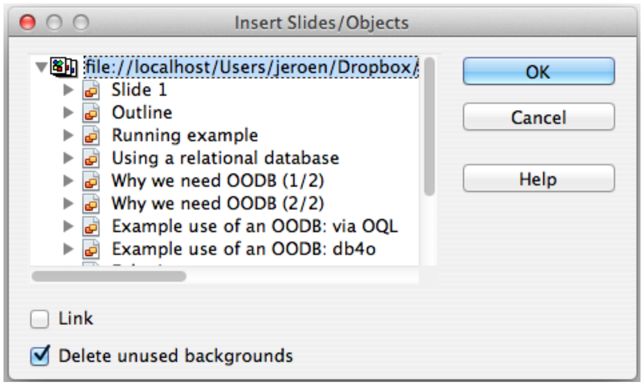

Insert Slides/Objects
file://localhost/Users/jeroen/Dropbox/
Slide 1
Outline
Running example
Using a relational database
Why we need OODB (1/2)
Why we need OODB (2/2)
Example use of an OODB: via OQL
Example use of an OODB: db4o
OK
Cancel
Help
Link
Delete unused backgrounds

Figure 2.14: Inserting a presentation in Impress results in the possibility to import the slides

(see Figure 2.9), it is not possible to import another Prezi presentation but it is possible to import a PowerPoint presentation. Doing this is not a perfect import however as a lot of colours and styling of the original slides is lost. When the user has dragged the slides from the sidebar into the Prezi presentation, they have to confirm that they want to add the slides on the spot where they dropped them. Instead of giving an annoying confirm dialogue, a small box is presented around the added slide with matching icons for confirming or cancelling the addition of the slides to the presentation.

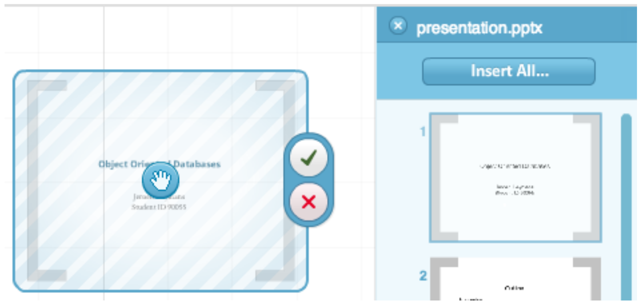


Figure 2.15: Importing a PowerPoint presentation in Prezi gives the user the possibility to import slide by slide

Conclusion There are a lot of differences in how the presentation tools handle the importing of slides. At first, there is Keynote which does not support it, then we have Impress which hides the feature from the user and is not user friendly at all. The tools that actually support importing slides via a menu option are PowerPoint and Prezi. Sadly, they are limited in terms of which slides they can import and the interfaces are not always user friendly.

2.1.1.5 Editing the structure of a presentation

The structure of a presentation is very important as it gives the presenter a basic order in which the content will be shown. In most tools this is a linear structure and Prezi is the exception in the tools that we compare. We compare the tools to see how easy it is to change the structure of the presentation. In most tools this will be a comparison to see how the user is able to change the order of the slides in the presentation.

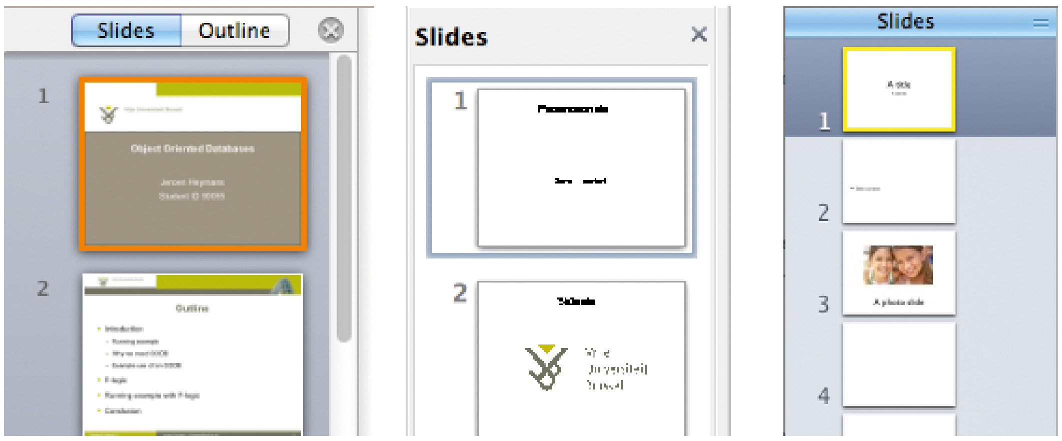


Figure 2.16: PowerPoint, Impress and Keynote offer the user an overview of his slides on the left side of the application. From L to R: PowerPoint, Impress and Keynote

PowerPoint In PowerPoint, an overview of the slides is available to the user. The user can choose between an overview of the slides themselves or the outline of the presentation. In both views, elements are moveable so the user can easily change the order of the slides. Depending on the width of the panel in which the overview is shown, the content is resized. A wide panel will result in wide previews of the slides.

Impress Impress offers a similar overview of the slides as in PowerPoint. It is however not possible to get an outline in the same panel. There is a possibility to get an outline but this is simply a different view to the presentation, which does not allow to change the order of the slides via this view. Impress is thus more limited than PowerPoint when it comes to changing the order of the slides in the presentation.

Keynote While in PowerPoint and Impress the previews of the slides resize according to the available width in the overview panel, this is not possible in Keynote as the panel is not resizeable. Keynote provides a very small selection menu at the bottom of the application to choose between small, medium and large previews. These are not sufficient enough to give the user a good overview of the presentation as even in the large setting, the text is barely readable. The outline of the presentation is just like in Impress not usable to change the order of presentations, the user is limited to the overview.

Prezi Prezi is quite different to the other presentation tools as it steps away from a linear structure and does not have the limitations of slides. A presentation is simply a collection of elements which can be connected via a path. But since this path is linear, Prezi gives a very similar overview of a presentation as can be seen in Figure 2.17. Via the `Edit path` button, the user can go into path editing mode where they can add new elements to the path and change it. If they do not click this button, the only thing they can edit from the path is the order in which elements are shown. This is similar as in the other tools where the user can move elements in the overview panel. This overview is necessary to give a better overview of how the path in the presentation goes. The reason behind this, is that in the presentation itself the path is only noted via numbers on the elements; the actual transitions are not visible.

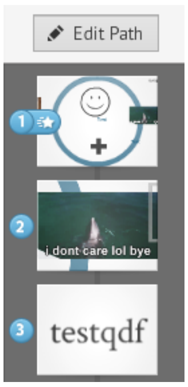


Figure 2.17: Overview of the presentation just like in the other tools

Conclusion As expected, PowerPoint, Impress and Keynote are very similar in how they give the user the possibility to change the order of slides: all have an overview panel in which the user can move the slides. The curious thing was that Prezi also provided this feature although it does not support slides. It simply shows the sequence of elements that are defined by the user in a path. The ordering of elements in the path is directly editable, if the user wants to edit the path in more detail, they have to go in edit path mode. A more consistent interface may be preferable. Also, the path itself is not that easy to see in a Prezi presentation when not in edit path mode.

2.1.1.6 Choosing a different theme for a presentation

When companies change name or their in-house style, the presentations all have to change too. This is only one example of where it is profitable if the user only has to change the theme of their presentations without having to edit them in detail. We compare the presentation tools to see how they implemented themes and how this affects the possible functionality to change the theme of a presentation.

PowerPoint As can be seen in Figure 2.18, choosing another theme in PowerPoint is easily done via the `Themes` tab in the ribbon which gives an overview of all the available themes. Sadly, there is no live preview available which forces the user to always select a theme to see what the result is.

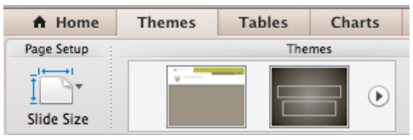


Figure 2.18: Choosing another theme in PowerPoint can be done via the ribbon

Impress Impress does not really implement themes as in PowerPoint which is very noticeable. Themes exist but in a very limited form. In order to apply a new theme, the user has two possibilities. The first one is that the user opens the Task pane after which they can select a new master for the slides. The other possibility is to select all the slides, right click and select `Slide design` in the menu to change the styling of all the slides. We note that this is very cumbersome and not user friendly as no live preview is available in both methods and the user cannot simply change all styling of all slides without selecting them all first or by making sure a very specific pane is visible to them.

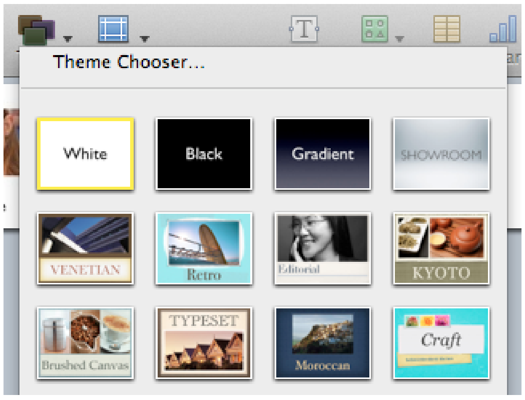


Figure 2.19: Choosing another theme in Keynote is relatively similar to PowerPoint

Keynote Changing the theme of a presentation in Keynote is very similar to changing the theme in PowerPoint as we see in Figure 2.19. Via an easy accessible button in the top of the application, a theme chooser appears in which theme thumbnails are shown. The user is obligated to click on them to see any changes in the presentation as there is no live preview available.

Prezi In the top part of the Prezi application there is a large button called *Template* and when clicking on it, the user sees a hovering menu that allows them to select another theme. This is very similar to how the choosing of a theme is done in Keynote. Unfortunately, no live preview is available but there is a *Revert to original* button which allows the user to revert back to the original theme that was set on the presentation.

Conclusion All the theme choosers are the same: they provide a thumbnail of the themes without live previewing. This is a missed opportunity as this forces the user to

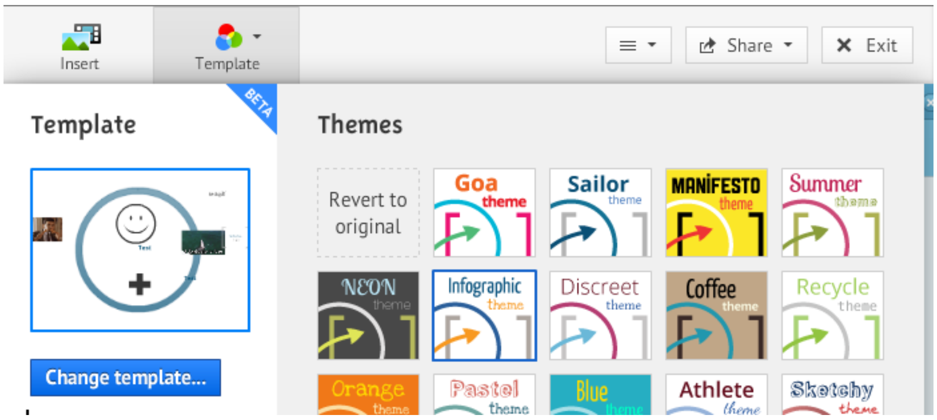


Figure 2.20: Choosing another theme in Prezi is very similar to Keynote

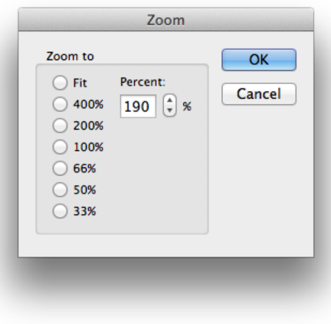
applying a theme and possibly redoing the process if they do not like it. Prezi is the only tool with an extra feature: the possibility to go back to the original theme of the presentation.

2.1.1.7 Zooming in a presentation

When editing a presentation, the user might want to zoom in to edit for example a shape more in detail. Or maybe they would like to have a better overview of the presentation to see the whole context of all the elements that are present in the presentation. We compare the possible zooming capabilities of the presentation tools as this is one of the major features of Prezi. We want to know how other tools have tackled the zooming feature.

PowerPoint Zooming in PowerPoint is possible via a small slider at the bottom of the application. This allows the user to easily zoom in and out. In order for the user to specifically say *Zoom to 150%*, the user has to go via the *View* menu at the top of the application where they will get the dialogue as shown in Figure 2.21 after some clicks. Although this gives us fine control over the zooming, it does not have a live preview function. The user can also zoom via scrolling if they hold the *Ctrl* key on their keyboard. Zooming is limited to the current slide that the user is viewing. If they want to zoom in the presentation, they can switch to the *slide sorter*. The *slide sorter* gives an overview of all the slides in the presentation. The user is able to zoom in this slide sorter which can help to grasp the overall structure of the presentation.

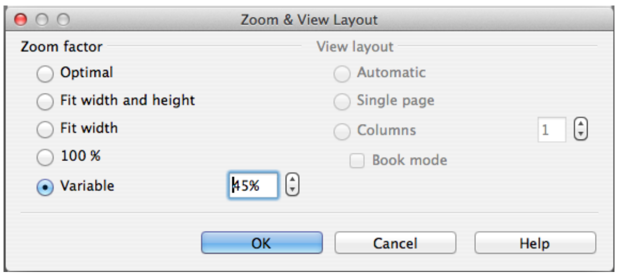
Impress Unlike in PowerPoint, there is no slider available to the user in Impress to zoom. The user either has to double click on the printed zoom percentage at the bottom of the application or they have to hold the *Cmd* key on their keyboard after which he can scroll to zoom. Just like in PowerPoint, there is no live preview available when the user zooms via the dialogue as shown in Figure 2.22. There is like in PowerPoint



Percent: 190 %

Figure 2.21: One of the possible interfaces to zoom in PowerPoint

a slide sorter available but in here the user cannot zoom. The slide sorter is thus only interesting to help the user with sorting slides.



[✓ Variable] 45%

Columns 1

Figure 2.22: One of the possible interfaces to zoom in Impress

Keynote Zooming in Keynote is even more limited than in Impress. There is only a small dropdown menu available at the bottom of the application with a few default percentages like *50%* and *150%*. The zooming is thus very limited for the user. Another problem in Keynote is that the zooming is limited per slide. There is no equivalent for the handy *slide sorter* like the one in PowerPoint which helps the user to better grasp the overall image of a presentation.

Prezi Zooming in Prezi works like a charm, just like we hoped. Simply by scrolling, the user's screen zooms. It zooms on the part of the presentation where the cursor is hovering, thus allowing zooming without having to pan to the correct position. It can be done in one fluid motion. There are also two small buttons available to zoom in and out if the user prefers clicking.

Conclusion We had high expectations for the zooming aspect in Prezi and had a feeling that zooming in the other tools would not be that great. As it turns out, we were correct. Out of the regular tools, PowerPoint has the best zooming capabilities. With the addition of live preview, this would be very powerful. Prezi excels in every aspect of the zooming and they have implemented the zoomable interface very well.

2.1.2 Theme editing

2.1.2.1 Editing the current theme of a presentation

If the user is editing a presentation and uses a theme, they might come to the conclusion that some things in the theme are not what they want. Maybe they want to have a bigger font size for the text or a different background colour. So we checked on how easy it is to start editing the current theme of a presentation.

PowerPoint When the user clicks on the tab *Themes* in the ribbon, they see the theme options like in Figure 2.23. Via this, it is very easy to change the theme colours, the theme fonts and the theme backgrounds of the currently active theme. If they want to edit more in detail, he has to start editing the *master slides* which allow a more fine-grained edit.

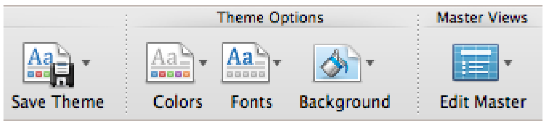


Figure 2.23: Editing a theme in PowerPoint is easily accessible via the Themes ribbon tab

Impress To start editing the current theme of a presentation in Impress, the user has to open the *master view* which is accessible via the *View* menu. It may be confusing to use the term *master pages* as there is no mention of an actual theme editor in Impress.

Keynote The user can directly edit the master pages of a presentation. To do this, they have to open the master pages pane which is not that trivial. It is done by reducing the size of the slides overview pane or via an option in the *View* menu. The result can be seen in Figure 2.24. Via this, the whole look of a presentation can be edited. The problem is that the word *theme* is never mentioned until the user tries to save it. This can make it very confusing for a user to know that they have to edit the master pages to change the actual theme of a presentation.

Prezi In Prezi there is a wizard available that helps the user to changing the definitions of their theme. In order to open this wizard, the user simply has to click on the button *Templates* after which the option *Customize Current Theme* allows them to edit the current theme that they are using in their presentation.

Conclusion Editing the current theme of a presentation is not always that easy. Prezi has the most obvious way to edit the current theme although it is still somewhat hidden. PowerPoint is the only one out of the remaining three presentation tools that allows the

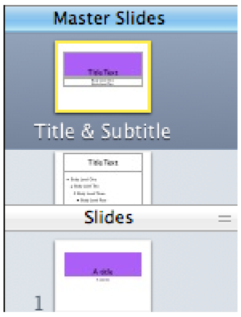
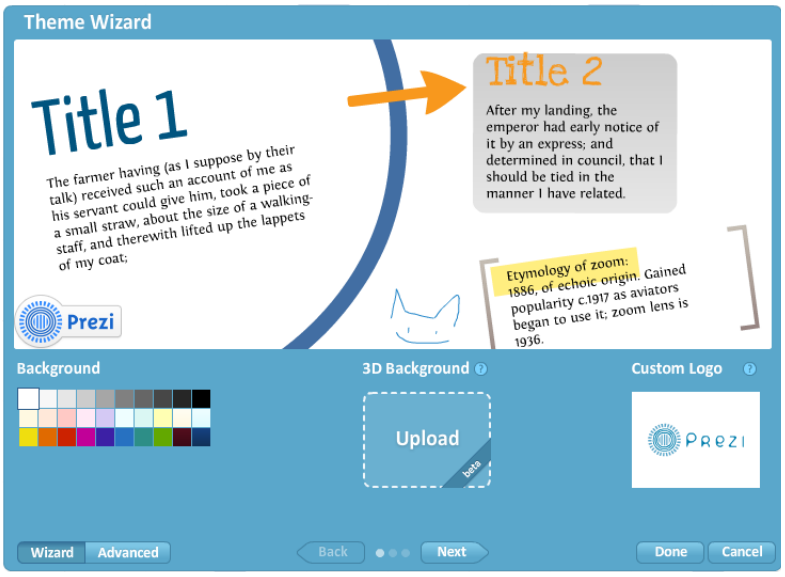


Figure 2.24: The master slides in Keynote are available in the pane above the slides



The farmer having (as I suppose by their talk) received such an account of me as his servant could give him, took a piece of a small straw, about the size of a walking-staff, and therewith lifted up the lappets of my coat;

After my landing, the emperor had early notice of it by an express; and determined in council, that I should be tied in the manner I have related.

Etymology of zoom: 1886, of echoic origin. Gained popularity c.1917 as aviators began to use it; zoom lens is 1936.

Figure 2.25: A wizard can guide the user into changing his theme

user to edit the theme on a higher level without requiring that the user edits the master pages for basic customisation like different colours.

2.1.2.2 How editing a theme affects presentations

We wanted to compare what the effects were on presentations that use themes. For instance, let us assume that we have a theme 1 and two presentations A and B. If both presentation A and B use theme 1 and theme 1 is changed while editing presentation A, is presentation B also changed? If yes, this means that presentations have a reference to their themes but if the answer is no, this means that theme settings are copied into presentations.

PowerPoint We noticed that in PowerPoint, themes are saved in separate files. When we replaced a theme file but kept the same name, nothing changed for the presentations that had that theme activated. The only thing that changed was in the overview of themes: there the thumbnail changed, suggesting that the presentation editor knew that the theme was changed. When looking further, we found out why the presentation editor knew that the theme was changed but why it did not alter the presentations. The reason is that presentations keep an internal copy of the theme that is currently applied. This can be seen in Figure 2.26 when changing theme: the first theme changes along when selecting another theme as it represents the current theme that is saved internally in the presentation.

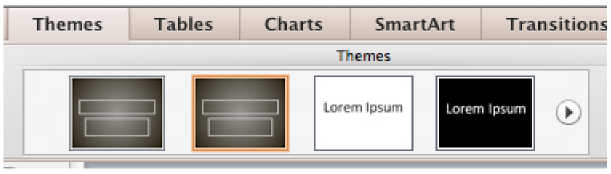


Figure 2.26: Every presentation seems to have their own theme

Impress As we saw in Section 2.1.1.6, Impress has its own complicated methods to choose the theme for a presentation. Changing a theme does not affect all presentations that use this theme. It is just like in PowerPoint: themes are copied into presentations so further changes to the original files are not visible.

Keynote The same story goes for Keynote as for PowerPoint and Impress: theme settings are saved in the presentations themselves.

Prezi Presentations are not affected in Prezi as it was impossible to overwrite a theme. Every time a theme was saved, it was added as a new theme. Thus when two presentations had the same theme applied, it had no effect when the theme in one presentation was changed as it was always saved as a brand new theme.

Conclusion In all cases, the changing of theme definitions never affected previous created presentations. This is on the one hand a good thing as this prevents the user from loosing possible custom styling definitions. On the other hand, if the user wants to change a theme that was applied to a bulk of presentations, they are forced to go over all their presentations and to select the new theme. By saving theme definitions in the presentations themselves, as done in PowerPoint, Impress and Keynote, a lot of duplicate information is created which could easily be avoided.

2.1.2.3 Changing the style of text elements

The styling of text is one of the many things that users may want to change in their themes. A bigger font size, a different font family and a different colour for the text are just a few examples.

PowerPoint As shown in Figure 2.23, changing things like the font families for a complete theme is very easy. When the user wants to have more control over what they change (for example when they want to combine multiple font families), they have to go into the master slides to change the individual appearance of elements. Although this gives them lots of freedom as the same editing features are offered as in the presentation editor, the user needs to know that they can edit themes on two levels. Either more abstract via the theme options where they can select from predefined possibilities or via the master slides editor which allows more fine-grained control.

Impress To change the styling of text elements in a theme, the user has to go into the master view in Impress. Via this, they have the regular controls to alter the settings for text, just like how they can change them in a presentation.

Keynote Changing the styling of text elements in a theme in Keynote is the same as when the user wants to edit the styling of text elements in a presentation. The only difference is that they have to switch to the master slides to make the changes in the theme itself.

Prezi In order to change the styling of text elements in Prezi, this has to happen via the wizard. In step 2, the user can define the styling for three elements: *title*, *subtitle* and *body*. The only thing that they can define is the font and the colour, both of which are fairly limited in choices as can be seen in Figure 2.27. While in the presentation editor the styling could be done in the presentation itself, the styling in a theme has a different interface in the form of a theme wizard.

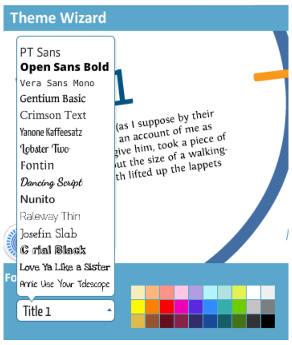


Figure 2.27: The styling of text in Prezi is done via the theme wizard

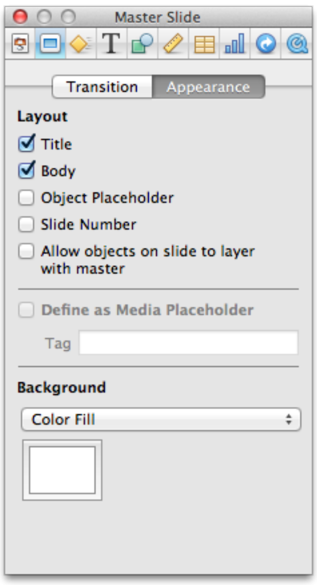
Conclusion Some presentation tools provide the same interface for styling text elements in the presentation editor as in the theme editor. However, in most cases the user has to go edit the master slides. This may be confusing as the concept of *theme* and *master slides* is not always that clear. Some tools like Impress even offer *templates* which makes it even more confusing. Still, as most presentation editors offer the same interface, it is not that hard to change the styling of text elements.

2.1.2.4 Changing the background of slides

PowerPoint As illustrated in Figure 2.23, the background of slides can be changed directly from the ribbon. The user can choose between default backgrounds, set some special gradient themselves or choose an image as a background. When doing this via the ribbon, the changes are applied to all types of slides in PowerPoint. For a more fine-grained control, the user has to go to the master slides editor where they can change the background for each type of slide.

Impress To change the background that is used in a theme, the user has to go to the *master view* and there they can change the background image just like during the editing of a presentation.

Keynote Changing the background used in a theme has to happen via the *master slides*. It is not as easy as one would hope. Firstly, the user has to have the inspector open after which he can go to the correct tab to select how the background should be filled as can be seen in Figure 2.28. The option is thus sort of hidden away and is not user friendly.



Layout
- ✓ Title
- ✓ Body
- ☐ Object Placeholder
- ☐ Slide Number
- ☐ Allow objects on slide to layer with master

- ☐ Define as Media Placeholder

Tag

Background

Color Fill

Figure 2.28: Changing the background has to go via the inspector

Prezi The changing of a background is similar to that of changing the styling of text elements in Prezi and can only be done via the theme wizard. It is also very limited at the moment. The user can only choose between different colours like can be seen for text elements in Figure 2.27. Images are not possible but they are currently working on the addition of 3D backgrounds.

Conclusion In all presentation tools, the changing of the background is similar as to how to change the styling of text elements. Some tools are sadly very limited or confusing. Prezi for example only allows a background colour while Keynote forces the user to go into a confusing interface called *inspector*.

2.1.3 Conclusion

We performed a comparison with four major and most popular tools: Microsoft PowerPoint, Apache OpenOffice.org Impress, Apple Keynote and Prezi. For each of these tools we always used the most recent version on Mac OS X. The reason for this, is that in this way we would be certain that the operating system was not a possible limiting factor in the features that the tools supported or in the interfaces that are available.

We asked ourselves several questions on how certain features are implemented in the current presentation tools. We could divide these questions in two categories: questions related to editing presentations and questions related to editing themes. In total we had seven questions for the presentation editing and four for the theme editing. We will now formulate our overall conclusions about the features that we investigated in these tools.

Presentation editing The support for different types of media depends greatly on the used presentation tool. While PowerPoint supports a whole lot of media types, Prezi is more limited. For example, Prezi only directly supports videos that are on YouTube. We could not find a way to directly add our own video file. This is not good as this forces a user to possibly choose a different presentation tool or to change their way of presenting certain information. The tool should never be a limit for the user. The user interfaces in the current presentation tools are either confusing or incomplete. We can conclude that no presentation tool is able to give the user a consistent interface to add new elements to the presentation that they are editing.

Moving elements in a presentation should be easy and straightforward as it is one of the basic operations that a user performs when editing a presentation. It turns out that there is quite a difference in how the four presentation tools tackled this problem. It is not possible to simple click anywhere on an element and drag it around in every tool. The sad conclusion is also that in all slide-based tools, dragging from slide A to slide B is not possible.

Resizing elements in a presentation only differs on how they handle the saving of the aspect ratio. All tools work intuitively enough to allow the user easy resizing of elements. Sadly, some limit the user in how he resizes images as they force to keep the aspect ratio.

Importing of slides is possible in all tools except for Keynote. We notice that all importing is a *by copy* operation, not *by reference*. We conclude that there is a duplication of information every time a user imports slides.

The editing of the structure of a presentation is generally not really a problem as most tools offer a slide overview or a path editor in the case of Prezi. The editing of the structure in slide-based presentations is only limited to changing the order of the slides. The path editor in Prezi allows a more flexible way to define and change the structure. The reason behind this, is that it works with a path that can connect different elements instead of only slides.

The biggest problem with choosing the theme for a presentation, is the lack of a live preview. Users always have to apply the theme after which they can evaluate if the change is what they want.

All presentation tools support a form of zooming when editing a presentation. Zooming comes in handy when the user wants to edit things in more detail like for example the resizing of a picture or aligning elements. Sadly, the zooming in some presentation tools is very bad or extremely limited.

Theme editing Editing the current theme when working in a presentation is not always possible. When it is possible, most presentation tools allow you to edit the theme that was copied into the presentation itself. This may be confusing for the user as the original theme files are not edited unless they overwrite them explicitly. In some cases like in Prezi, it even is not possible to overwrite old themes.

This has its effect on how presentations react when a theme is changed. Because of the internal copy of the theme, no presentation has a changed appearance. This may not be interesting for the user. For example in a company, it is possible to have a general theme for all the presentations. But if the company changes name or look, the presentations will not be updated unless the user walks over them individually to update the theme that is used in the presentation.

Changing the style of text elements and the background of slides could in general be achieved by using the same interface as when editing the presentation. This is good as it gives a certain consistency but sadly, most of the actual editing could only be started after going through some menu's and dialogues. A good example of this is the *inspector* dialogue in Keynote and the use of *master slides* or *master pages* in other presentation tools.

Overall conclusion Some features were fairly good and intuitively implemented in certain presentation tools. There are however a lot of features that are very poorly implemented. Either these features are hidden very deeply away in the interface, they have inconsistent naming or they are very limited in the options that they provide. A more extensible and consistent interface would be very beneficial for most of the tools.

2.2 Current available technologies

Most of the presentation tools mentioned in section 1.1 are built in languages like C++, C# and so on. The only exception is Prezi which is more webbased. However, it extensively uses Flash which requires a browser plugin. The easy choice for developing a cross platform application would be to look at a language like Java which is intended to let application developers work according to the WORA principle: write once, run anywhere. However, this means that the user still has to have Java installed on his machine. What if we want our application to run on mobile phones and tablets, which do not always support Java?

An alternative is to choose web technologies as a basis for our application. More than ten years ago, it was already predicted that web browsers would become a widely used user interface [61]. If we will be building the presentation tool for use in a web browser, this means that it would have to be programmed in HTML, CSS and JavaScript. As the web is constantly evolving, the choice to go for the latest technologies is trivial: HTML5 and CSS3.

HTML5 [1] is the successor of HTML 4.01. Currently, HTML5 is a work in progress by the W3C and is expected to go into *Recommendation status* in 2014 [2]. HTML5 contains new versions of XHTML 1 and the DOM2 HTML API [3]. Previously, these were defined in separate specifications but now they are merged in one standard.

With the arrival of HTML5, new elements and attributes were introduced and some elements and attributes were removed [72]. Along with these changes, other elements got new semantics. Although people were critical about the adaptation rate of browsers, support for HTML5 and CSS3 is constantly growing and the popularity keeps rising. According to a survey by VisionMobile, more than 50% of mobile developers already use HTML5 [73]. As developers start to use HTML5 and CSS3 more and more, browsers have to follow.

While HTML5 provides new features like a video tag and the ability to have client-side databases; CSS3 provides features like animations and transitions [68]. However, one has to rely on JavaScript for dynamic scripting. JavaScript has been pushed forward as the best language to create rich web applications [56, 51]. Some even see a future where web browsers become a dominant client application platform [74]. With this in mind, the choice to use JavaScript was evident.

However, as a presentation tool is not a small job and generally offers a lot of functionality, we can expect to have a very huge amount of code. The use of libraries thus will be necessary to avoid having to reimplement basic stuff that we want. Some libraries have very specific purposes as they provide for example functions to manage XML files.

Some libraries like jQuery, Dojo and so on are targeted to use as a basis for a web

[1]`http://www.w3.org/TR/html51/`
[2]`http://dev.w3.org/html5/decision-policy/html5-2014-plan.html`
[3]`http://www.w3.org/DOM/`

application. Thus, we decided to start a JavaScript library comparison to see which library we would use as a basis. This comparison can be found in section 4.1. In this, we did a feature comparison to see what they promise to have available. We also did a comparison by implementing some basic features with the help of multiple JavaScript libraries.

2.3 Use of interface languages

A User Interface Description Language (UIDL) is a high-level language that is used to describe a user interface. There are a lot of languages out there and we wanted to see if we could use one of them. Because MindXpres is plugin-based, it could be interesting to see whether the plugins could use a certain interface language.

The reason behind this, is that it gives a certain abstraction. It allows developers of plugins to define aspects of the interface without having to know how the application works under the hood. For example some user interface languages allow the definition of what elements should be present. The WYSIWYG editor that will be developed in this thesis could render these definitions to get a uniform look. Another possibility is that the user interface language gives developers complete freedom to define what and how elements should be displayed to the user.

Another useful use of an interface language is based on the fact that different users of the system have different needs. By abstracting the actual rendering of the interface and using an user interface language, it is possible to take certain preferences into account during the actual rendering process. For instance if user X prefers bigger text than user Y, he could put this in a preference. When the translation happens from the user interface language used in the plugin to for example HTML5 and CSS3, the resulting HTML5 and CSS3 could be dynamically changed to fit the preferences of user X.

We compared several user interface languages to identify what is out there, what were possible candidates and what would be the advantages or disadvantages of using them.

2.3.1 Interesting user interface languages

UIML UIML stands for User Interface Markup Language and is an XML-based language that provides a declarative syntax for developers to build user interfaces [1]. Instead of defining how a user interface has to look, UIML allows to define what the elements in the interface are. Via a renderer, the user interface is generated with a uniform look.

UsiXML Another language we looked at was UsiXML [46]. It is in fact a specification language that allows the designer to describe a user interface on different levels of abstraction. A designer can describe at a high level of abstraction the constituting elements of an user interface (e.g. widgets, containers, modalities or controls). It supports device, platform and modality independence.

XAML XAML stands for Extensible Application Markup Language and is a declarative markup language developed by Microsoft [48]. It allows developers to define UI elements, bind data, handle events and so on. XAML is mainly used in the .NET framework for desktop applications.

XForms XForms is an XML language that is used to specify how data must be processed [14] such as is the case in web forms. Originally, XForms was designed to be the next generation of (X)HTML forms. Now however, it is more generic and it can be used with other markup languages than XHTML.

XUL XUL is a user interface language developed by the Mozilla Project [25]. XUL stands for XML User Interface Language and is an XML dialect. It allows developers to write graphical user interfaces in a similar fashion as HTML is used for web pages. The XUL is interpreted by a layout engine, allowing for cross-platform applications.

2.3.2 Pros and cons of the user interface languages

We will now evaluate the different user interface languages that we presented in the previous section. What are the positive and negative aspects of the languages?

UIML UIML uses the declarative syntax as illustrated in Listing 2.1. In the description tag, elements can be defined. Via the structure tag, we can define how the previously defined elements are ordered and organised in a structure. The other tags are quite trivial: the `data` tag allows the injection of data, the `style` tag allows the developer to define possible styling and the `events` tag lets the developer define on what events elements should react.

```xml
<?xml version="1.0" standalone="no"?>
<uiml version="2.0">
  <interface name="aName" class="aClassName">
    <description>...</description>
    <structure>...</structure>
    <data>...</data>
    <style>...</style>
    <events>...</events>
  </interface>
  <logic>...</logic>
</uiml>
```

Listing 2.1: A basic UIML file

A huge pro for using UIML is the complete separation of concerns. A drawback for using UIML in a rich Internet application is the lack of a renderer in JavaScript. At the time of writing this thesis, we could only find an implementation of a renderer in Java:

jUIML[1] and one in C#[2]. To get UIML working in a rich Internet application, one would have to write a complete renderer themselves.

UsiXML The biggest pro for UsiXML is the fact that there are multiple levels of abstraction possible to define elements for the user interface. To support the editing of these multiple levels, the UsiXML consortium provides different tools[3]:

- Task model editor
- Domain model editor
- Context model editor
- Abstract User Interface editor
- Concrete User Interface model editor

The problem is that these tools are not freely available. To fully support UsiXML, one has to make the cost of buying tools.

XAML Although XAML is mainly used in the .NET framework, Microsoft allows developers to use it in a browser. This can be achieved via XAML Browser Applications which are complied applications that run in the browser. To get this working, WPF needs to be installed. Another method is by using the Silverlight plugin. We can clearly spot the problem: XAML will only work via external tools. These external tools might work on desktop computers like the Silverlight plugin but this makes the application rely on external applications.

XForms XForms is the final user interface language we looked at. Like the other user interface languages, there is the need for a processor to generate HTML that can be used by the browser to display the page. In contrast to other user interface languages, XForms has an implementation that can run entirely in JavaScript and does not need any plugins. It is called XSLTForms[4] since it uses a combination of JavaScript and XSLT. The biggest drawback for XForms is the lack of alternatives. XSLTForms is only maintained at the moment by one person and is still in a release candidate phase[5]. At the moment, no big browser supports the rendering of XForms natively. Firefox even dropped support in version 19[6].

[1]`http://sourceforge.net/projects/juiml/`
[2]`http://research.edm.uhasselt.be/kris/projects/uiml.net/`
[3]`http://www.usixml.eu/usixml_tools`
[4]`http://www.agencexml.com/xsltforms`
[5]`http://sourceforge.net/projects/xsltforms/files/xsltforms/`
[6]`https://developer.mozilla.org/en/docs/XForms`

2.3.3 Choosing a user interface language

When looking at the analysis of all the user interface languages that looked promising, we can spot a clear trend: there is almost no support to render the languages to HTML. To solve this problem, one would have to write a render themselves or make use of technologies like XSLT. This is however very problematic as it involves an enormous amount of work. The best example can be found in XSLTForms which is a one man project that is going on for almost 5 years and still has not reached its first version.

We conclude that no user interface language is usable on a short-term basis. On the long term, all of them have their benefits and practical uses. Since one would have to write a renderer themselves, it would be a matter of personal preference which language to use.

3 Designing a better presentation tool

Designing the best possible presentation and theme authoring tool for MindXpres [62] is not an easy task. Presentation tools are traditionally very big and complex because of the enormous amount of functionality. The most common presentation tools have been in development for many years by big companies. In the previous chapter we identified a list of problems that are currently present in presentation tools. In this chapter, we will introduce our design for a good presentation tool by improving certain concepts that we identified as problematic. We also introduce some novel ones that are currently unavailable in presentation tools. We designed several mockups to visually support our ideas.

3.1 Type of interface

A first question we asked ourselves was: how to let the user edit their presentations? MindXpres presentations are targeted to be used with a zoomable user interface while presenting. But does this mean that the editor should also have a zoomable user interface?

ZUIs have been presented as an alternative medium for slide show presentations [35]. They enable distinguished levels of detail, spatial navigation, paths and an alternative for slide transitions as the user traverses the presentation through a multi-scale 2D space. In the existing tools the user often makes the mental separation between their presentations at authoring time and the final result. This is partly aided by the fact that the two versions are sometimes presented very differently. In our tool, we want the presentation to be as close to the final result as possible while authoring. ZUIs are proven to be helpful in presenting hierarchical diagrams [30]. As shown by the iMapping tool [36], ZUIs allow an efficient visualization of hypertext. Because a MindXpres presentation can be thought of as a hierarchical diagram and it is based on hypermedia, the choice for a zoomable user interface is well-reasoned. In the following section, we discuss some of the benefits of ZUIs.

3.2 Solving the problems with Zoomable User Interfaces

Zoomable user interfaces have several known problems when using them in applications: the lack of context, the negative effects of zooming on the perception of the user and excessive animation. Firstly, we try to tackle these problems and then we will give guidelines for how possible solutions can be used in a presentation tool context.

3.2.1 Lack of context

One of the biggest problems with ZUIs is the lack of context. In research, context aids have been developed like hierarchy trees [60], flip zooming [11], history layers [60], focus-and-context [16, 17], collapse-to-zoom [4] and detail+overview [17].

Hierarchy trees Hierarchy trees are a proposed technique to give the user a view of a flattened vertical slice through the information space that they are navigating. A presentation in MindXpres can be seen as a graph with possible multiple levels of detail. The user generally starts at the top with the least amount of detail visible and as they zoom in, more details are shown. The objects in a presentation can have other objects in them that provide more detail and thus are only visible after zooming in. A hierarchy is thus indirectly created. A hierarchy tree is a technique where the user gets a flat view of the hierarchy they are currently browsing, thus giving them an idea of what the context is. Although this information of the context is what we want to achieve, the problem is that MindXpres presentations are not always representable in a deep hierarchy. For instance, it is possible for the user to create a flat presentation with all elements on the same level. In such a presentation, there is no hierarchy which would make hierarchy trees useless.

Flip zooming In short, flip zooming is a technique that presents discrete and sequential information in a number of tiles. At any given time, one tile is focused on and gives the user more details. The other tiles are sorted around the focused tile to give the user an idea of what the context is of the details that he is looking at. The problem with this technique is that it only works on discrete and sequential information. Although the objects in a MindXpres presentation are discrete, they are not ordered in a sequential fashion. We conclude that flip zooming is not usable for the ideal presentation tool as it limits the flexibility.

History layers The lack of context in a zoomable user interface leads to questions from the user like: *Where am I?* and *How did I get here?*. By default, ZUIs do not have a history so the user cannot go back in time. By going back in time, we mean that the interface jumps back to positions where the user viewed a part of the presentation in the recent past. By adding a history layer [60], the user can view where he was for example five minutes ago. This way, he can go through a presentation and then see where he was to better grasp the whole context.

Focus-and-context The idea behind focus-and-context is that users get to see an object in full detail while the surroundings are viewed in much less detail. An example of a focus-and-context technique are Fisheye Views [26]. With Fisheye Views, the user utilises a lens that magnifies the center of the field of view. The magnification goes down towards the edges, thus effectively generating a fisheye view. As MindXpres uses ZUIs without Fisheye Views, it is not that interesting as it changes the whole perspective for the user: he will get another view of a presentation and so the WYSIWYG-editor would not be really WYSIWYG.

Detail+overview Detail+overview interfaces are very promising as they offer the user an overview of where they are currently navigating in the interface while displaying a more detailed view alongside it. Studies have shown that it is mainly a subjective preference of the users that may lead to higher usability [37] but that it is also effective for the user to determine where different objects are located with respect to each other [12].

Conclusion Because of the problems, limitations and shortcomings of the other techniques, we propose to use detail+overview. A small overview window is shown where a box displays what portion is rendered in the detailed part of the screen. This can be combined with history layers that allow the user to go back in time in their navigation.

3.2.2 The effects of zooming on the perception

Another problem with ZUIs is how the actual zooming affects a user's perception and usability. To achieve better usability, novel things have been introduced, including speed-dependent automatic zooming [40] in which the system zooms out when the scrolling speed increases and will zoom back in when the scrolling speed decreases. When evaluating this speed-dependent automatic zooming technique, a performance gain of 43% could be achieved [18].

3.2.3 Excessive animation

Although animation can help users build and reconstruct a good mental map of spatial information [9], caution is necessary to make sure not too much animation is used. This could influence the usability negatively. Animation abstractions have been described in literature [39] to give a guideline what kind of animation techniques are interesting: motion-blur, squash-and-stretch, use of arcing trajectories and so on. We propose to use these techniques for when the user clicks in the overview window, to let the detailed view change dynamically without letting the user loose his feel for the context. The user performance for decision making is higly contingent on things like how smooth animations are [33]. Use of animation is recommended but careful consideration is necessary on how it is implemented, user tests are advised.

3.3 How presentations and themes are linked

In Section 2.1.2.2 we noted that in current presentation tools, presentations hold an internal copy of their theme. This is very bad practice as it gives rise to two problems: duplication of data and that changes in themes do not propagate to the presentations that use the changed theme.

According to Vannevar Bush [13], this duplication of data occurs because computers often use hierarchies for storing information, from the filesystem level down to the document format level. Such a hierarchical structure does not allow loops to be formed but these are needed for inclusion by reference. In hypertext systems however, we finally step away from the hierarchical representation of data and links between documents are multidirectional.

Ted Nelson has described the concept of *transclusion* or *inclusion by reference* in his articles about hypermedia [55]. What this means is that (possible external) content is included at viewtime. An easy example to show how this works is in the context of an

HTML document. This document can have image tags which may link to an image on the server, on the computer of the user or even somewhere else on the World Wide Web. It is only at viewtime that the image is resolved and shown to the user as if it were part of the HTML document. In a hypertext system, this is not limited to images only but can be used for any kind of media.

We propose to use themes and presentations in a hypertext context. Themes should be transcluded into presentations and themes could even transclude other themes. This would result in a hierarchy of theme definitions which solves the problem mentioned in Section 2.1.2.2: a presentation's look was not updated when the original theme was changed. By using transclusion, themes would be resolved at viewtime. This effectively allows the user to have themes and presentations completely separated. It gives the user the default ability to centralise all theme definitions in themes without having to worry about possible duplicates in the presentations.

Figure 3.1 shows how themes and presentations are linked together. Presentations themselves can have custom styling on their elements. This gives more freedom to the user as they can apply a theme to a presentation while customising certain elements to their own taste. Styling definitions thus occur on two levels: on the level of the theme and on the level of the presentation. As themes can have a base theme, a hierarchy of styling definitions can be made. The priorities of the styling definitions can be ranked from lowest to highest:

1. Base theme of the theme
2. Theme of the presentation
3. Custom styling definitions in the presentation

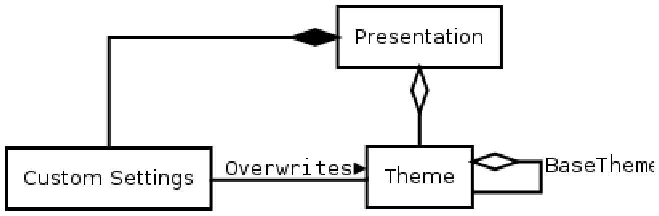


Figure 3.1: A presentation has a theme which on its own can also have a base theme. Presentations have their own custom settings which override the theme settings.

3.4 Presentation and theme editors: separated or not?

The next question we asked ourselves was, How do we allow the user to edit his presentations and his themes in the same program?' As we have seen in Sections 2.1.2.3 and 2.1.2.4, lots of current presentation tools offer a very similar interface to edit the presentations and the themes. In fact, most of the interfaces are reused as most tools work with the concept of *master slides*. Therefore, it looks to the user that they are editing a presentation while they are actually editing a theme.

It would be much better if the user could clearly see that they are editing a theme. This is why we propose to have the presentation editing and the theme editing separated into two different editors. This gives rise to a new problem: how do we link these editors together? For example, how do we allow the user to edit the current theme of their presentation. Potential solutions that are used in current tools are presented in Section 2.1.2.1.

Another question is of course, how do we let the user go in the other direction? What if they create a theme and decides to directly start building presentations based on that theme? Or what if they want to apply a theme to a presentation without having to bother with opening the presentation? In current tools, we could not find the support for such scenarios.

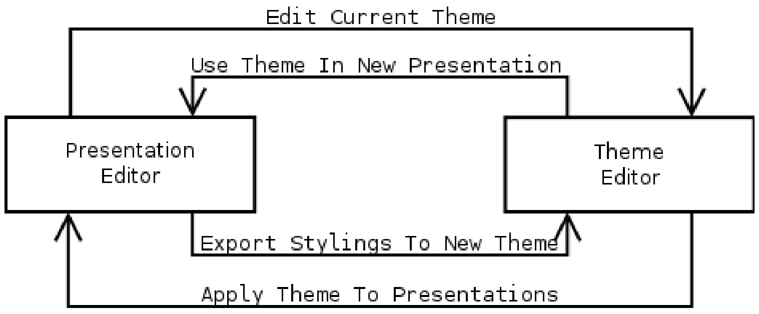

Edit Current Theme
Use Theme In New Presentation
Presentation Editor
Theme Editor
Export Stylings To New Theme
Apply Theme To Presentations

Figure 3.2: Four connections are made between the presentation and theme editor for maximum flexibility

We propose to have two separate editors: one for presentation editing and one for theme editing. But to have a bigger amount of usability, the two editors are tied together in different ways as illustrated in Figure 3.2. We will go over the different links, explaining what their purpose are.

3.4.1 Editing the current theme

This is a feature that we compared with the current tools is section 2.1.2.1. It turns out that most of the current presentation tools support the editing of the current theme but this is not always that clear to the user. We propose a solution where there is an interface element like a button available to the user to edit the current theme of the presentation. Via this interface element, they switch to the theme editor to directly edit the theme. When they save, the theme is updated in the presentation itself without a problem thanks to the transclusion that we proposed in Section 3.3.

3.4.2 Using a theme in a new presentation

A feature that we missed in current presentation tools, is the ability to create a theme without having to create a presentation and to create a new presentation afterwards with that theme. This may sound like making life too difficult as a user can create the new presentation already and edit the theme afterwards. The problem with this method in current presentation tools is that a theme is edited in a presentation. Each presentation holds in fact a copy of the original theme definitions. This is why we identified in Section 2.1.2.2 that other presentations were not affected when editing a theme. The user is obligated to save the theme afterwards in its own theme file, effectively having duplicate theme definitions as the presentation itself also holds a copy.

By decoupling the theme definitions from the presentation, we allow the user to create a theme separately without having to bother in which presentation it might be applied. The *Use theme in a new presentation* functionality then simply acts as a shortcut for closing the theme editor and creating a new presentation where the user has to go select the recently created theme. Via the shortcut that we propose, the theme is already selected to be applied in the new presentation that will be created.

3.4.3 Exporting stylings to a new theme

In current presentation tools, a theme is in fact imported into a presentation as all the theme definitions are duplicated. But what if we want to go the other way, if we want to export a theme out of a presentation? As a presentation is created, certain minor tweaks to the settings may be done to the presentation itself: different font family, different colours and so on. These tweaks may reoccur in other presentations.

To reduce the amount of duplicate data, it would be better to save all these new settings in a theme. By providing the *Export stylings to a new theme* functionality, the user is able to generate a theme based on the custom settings he has made in a presentation. What in fact happens, is that a reference to the original theme of the presentation is saved and that the custom settings are then added to the new theme as illustrated in Figure 3.3.

3.4.4 Applying theme to presentations

The final connector between the theme and presentation editors is the *Apply theme to presentations functionality*. One of the biggest issues in current tools is that editing a theme does not affect the presentations that utilise this theme as we have seen in Section 2.1.2.2. A user is thus forced to open all their presentations and change the theme. If we completely decouple the themes from the presentations and if there is no internal version of the theme like in current tools, this problem will not occur.

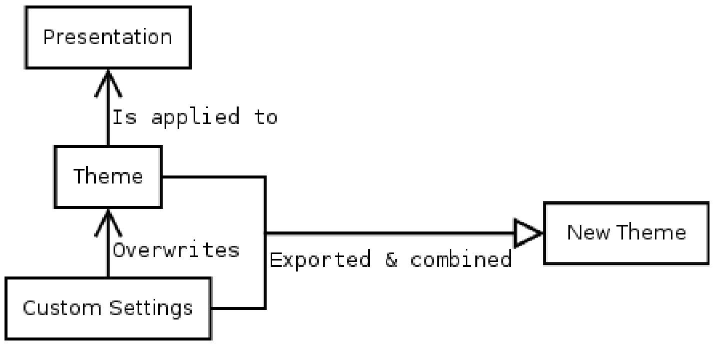


Figure 3.3: Exporting a theme is in fact the combination of custom settings and the original theme

Still, it is preferable to give the user the freedom to select another theme for a presentation. But what if the user creates a new theme and wants to apply it to multiple presentations? In current tools, they have to go over all the presentations to change the theme that is applied. We propose a functionality where the user is able to simply apply a theme to multiple presentations without having the need to open them. This is for example ideal if they have multiple themes where only the colours are different. This does not affect the presentations in a critical way so they do not need to go over each of them to verify if the change of the theme does not break the presentations in any way.

3.5 The presentation editor

3.5.1 The interface

As can be seen in Figure 3.4, we propose a minimalistic interface. One of the biggest issues in current presentation tools, is that there are multiple ways of doing certain actions. Although different methods can be good, they do not always support the same set of features as we could clearly identify in Section 2.1.1.1. We propose to have one big menu that allows the user to fully control the presentation editing, tabs under the menu and a sidebar that is built out of two parts: a navigation panel and a basic overview panel.

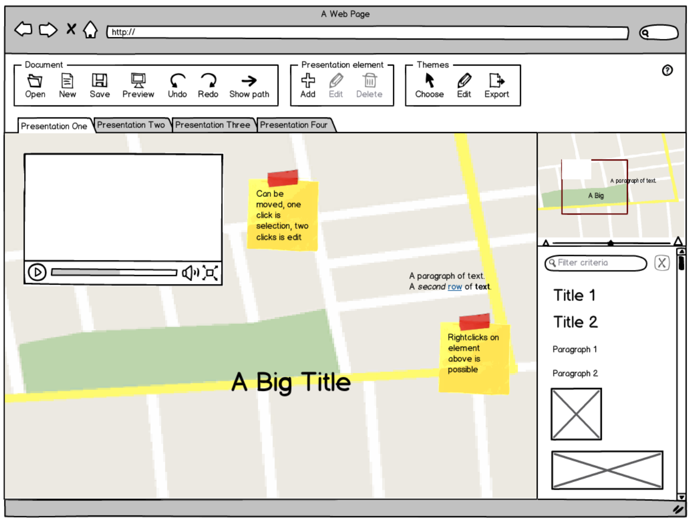


Figure 3.4: An example view of when a user has a presentation opened

The menu itself should be horizontal with different groups that are logically chosen. The first one is a more global group for the editing of presentations in general. This group allows opening and creating a presentation but also allows the user to do certain global actions on the presentation they are currently editing: undoing an action, redoing an action, showing a preview of the presentation and showing the path in the presentation. Another group is the one that allows the authoring of the presentation elements: adding, editing and deleting elements. The final group concerns the link with themes. It gives the user the possibility to choose a base theme for his presentation but also allows the user to jump to the theme editor. This jump to the theme editor can be achieved by wanting to edit the current theme of the presentation or by exporting a new theme based on the currently opened presentation which we discuss in Section 3.5.10.

The tabs under the menu are available to the user so they can easily switch between editing multiple presentations. Thus it is possible for the user to have multiple presentations opened at the same time. When all presentations are closed, they should automatically return to the starting screen as shown in Figure 3.5.

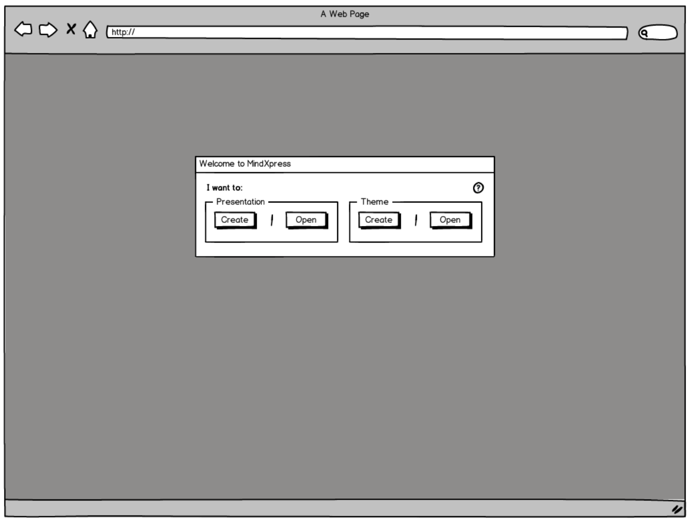


Figure 3.5: The screen that the user sees when opening MindXpres

The navigation panel allows the user to navigate in the presentation. There we show an overview of the presentation. Although it is proven that interfaces with overview+detail do not affect the performance for navigating with a zoomable user interface, they are found to be more enjoyable and physically less demanding [53]. It is also shown that users prefer overview+detail in electronic documents [38]. Therefore we decided to propose the use of an overview+detail interface with the detailed view in the sidebar. This will possibly increase the contentment of the user. We allow the user to navigate in the presentation by manipulating the detailed view by which a performance gain can be achieved [5].

The basic overview panel is the final panel in the interface. It allows the user to see all the elements that are in a presentation, combined with a search functionality. The idea is that it actually works as a kind of filter. By default, all the elements in the presentation are listed and while the user types in the text field, the elements get filtered. As they are filtered and the list shrinks in size, the matching elements are also shown in the presentation itself. The reason why we also show them in the presentation is that context helps users to find specific elements more easily [22].

To highlight the matching elements in the presentation itself, we base ourselves on three principles. The first one is that colours that are contrasting to their surroundings draw more attention as they do not blend in [63]. The second one is that large objects draw more attention to them [70]. The third and final one is that appearing objects draw attention [75]. Based on these three principles, we propose to highlight the matching elements in a contrasting colour with a bigger font and with a small animation to make them appear.

An extra feature of the overview panel is that elements are clickable. When the user clicks on an element, the presentation editor jumps to the part of the presentation that contains that specific element. It is thus an alternative way of navigating in the presentation.

3.5.2 Creating a presentation

Creating a presentation is possible via the start screen or in the menu at the top of the application while the user is in the presentation editor. To create a presentation, only basic information is required as can be seen in Figure 3.6. The user is able to select the theme that he wants to use and can provide information like his name.

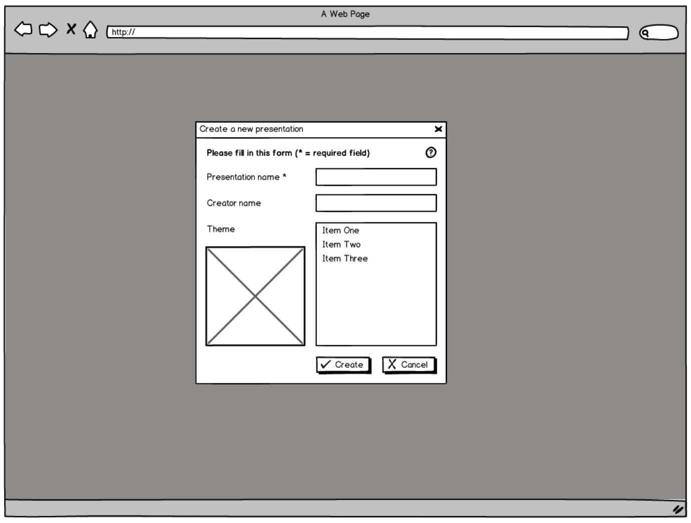


Figure 3.6: To create a presentation, only basic info is needed

3.5.3 Adding elements

As shown in Section 2.1.1.1, adding elements in current presentation tools can be quite counter-intuitive. Thus we propose one single menu to add elements as shown in Figure 3.7. The problem with having only one menu to add elements, is that the menu can become very crowded. To address this, we propose to have a filter mechanism on the top of the menu. When the user starts typing, the elements of the *Add element* menu are filtered. The result is that when the user types in the keyword *text*, for example only the elements *title* and *paragraph* will appear.

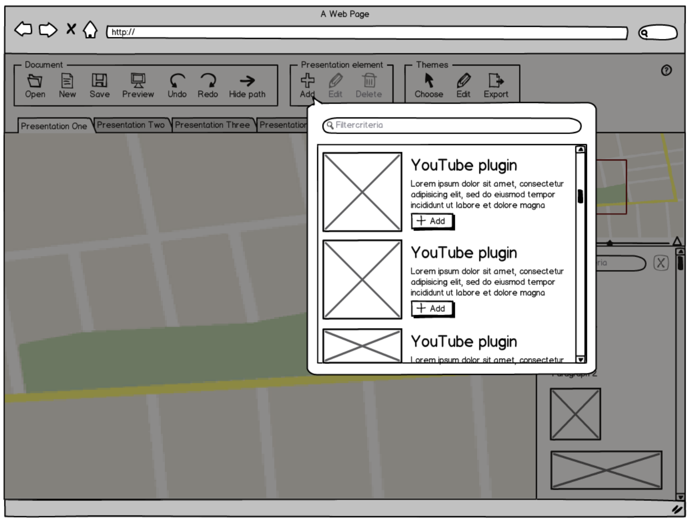


Figure 3.7: The menu to add new elements to the presentation

Another possibility to address the problem of a very big menu is to use fisheye menus [7] which are a very promising technique for when a menu gets to 100 elements and when creating a hierarchy in the menu is not really an option.

3.5.4 Changing the settings of elements

After the user has added elements, they should be able to easily edit the settings of the elements. Via a dynamic dialogue, the user should be able to preview their changes that they perform. An example of such a preview can be seen in Figure 3.8.

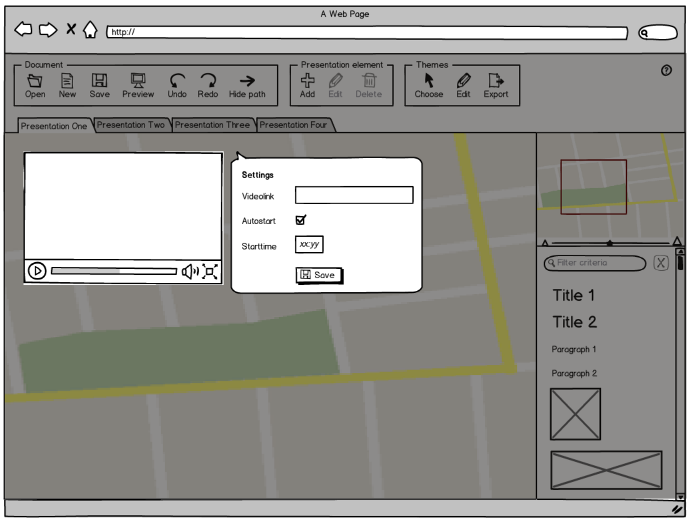


Figure 3.8: Editing the settings of a presentation element

The changing of settings can be accessed by right clicking on an element. Right clicking triggers a context menu as illustrated in Figure 3.9.

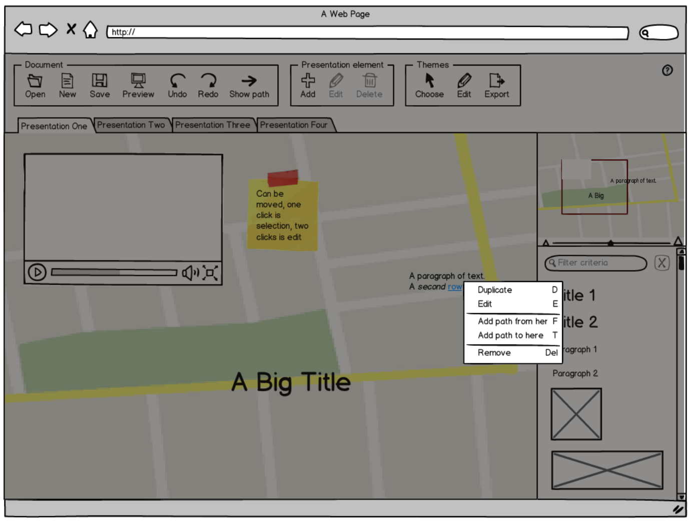


Figure 3.9: A context menu that allows easy access to operations that the user can perform

Right clicking outside of a presentation element should also trigger a context menu which gives access to different actions as shown in Figure 3.10.

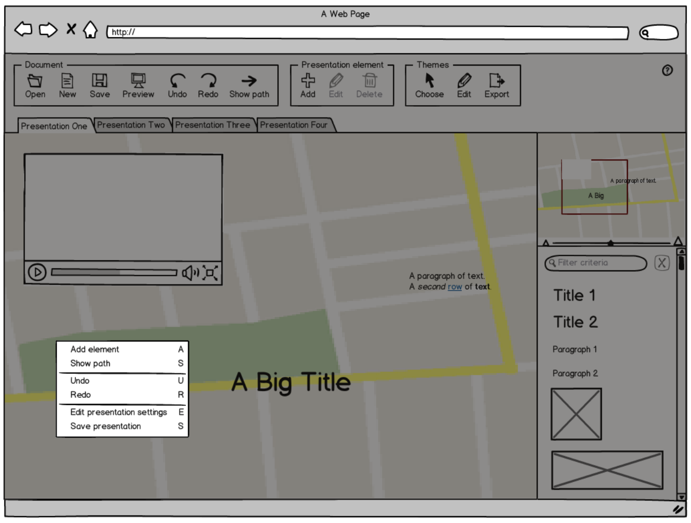


Figure 3.10: Offers shortcuts to user in the presentation editor

3.5.5 Moving elements

Moving elements should be very intuitive. As we saw in Section 2.1.1.2, certain presentation tools have the problem of having a dead space on elements. For example text elements could only be moved when clicking on the actual text or the border of the text element. We want to avoid this unreasonable behaviour by simply allowing the user to move an element by clicking anywhere they want. A hint should also be shown so the user knows on which element they are hovering. This can be in the form of a changing cursor but also a bounding box around the hovered element.

3.5.6 Resizing elements

As we have seen in Section 2.1.1.3, the current presentation tools offer two types of resizing, with or without default saving of the aspect ratio. As the default saving of the aspect ratio is a limitation to how the user can use the tool, we propose to have the most flexible type of resizing. Each element should be resizable in eight directions as can be seen in Figure 3.11 where we have drawn the possible points to allow resizing on an element. By default, the resizing does affect the aspect ratio. If the user wants to preserve the aspect ratio, they can do this by holding the *Shift* key on their keyboard.

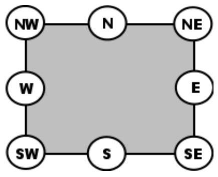


Figure 3.11: The eight possible directions in which the user can resize an element

3.5.7 Editing the structure

In Section 2.1.1.5 we identified that most presentation tools have a linear structure which can be edited by simply changing the order of the slides. Prezi was an exception to this as it offers the user to draw a path. The actual path could also be edited in a similar fashion with the presentation tools that offer a linear structure. The reason is that Prezi presentations have a linear path. For instance, it is not possible to have two edges starting in the same element that would eventually allow the presenter to choose which path he will follow during his presentation.

A first proposal for editing the structure, is that the currently defined path can easily be shown in the presentation as shown in Figure 3.12. When using Prezi while in the *edit path mode*, the path is shown in a separate panel but not in the presentation itself. There are hovering numbers to show the user the sequence of the elements in the path but it would be much more user-friendly if the user sees the actual edges. This comes in handy when the user is, for example, zoomed in a lot so he can clearly see where the edges are going to and coming from. It increases their spatial awareness as not all numbers may be visible when zoomed in.

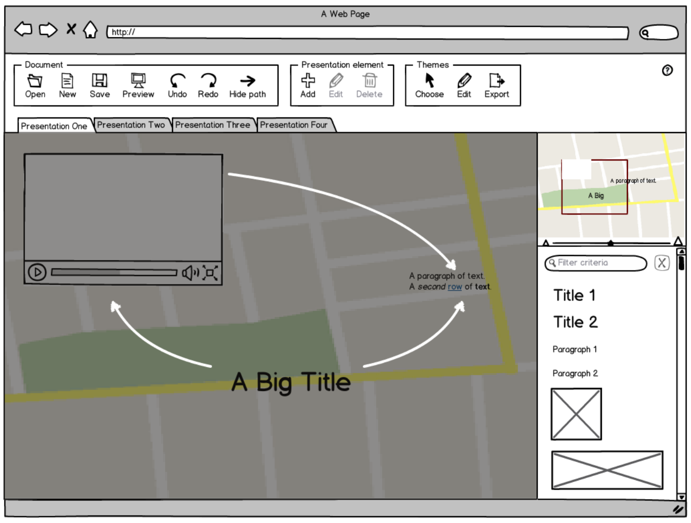


Figure 3.12: Clicking on Show Path triggers this view where the user can clearly see the path

We propose to have a path editor like in Prezi but without the limitations that we mentioned. It should be possible for the user to draw edges between any two elements and it should not matter how many edges already depart from the same element. The path editor should be totally flexible. Edges should be moveable: their target and origin should easily be changed, preferably by simply dragging them to another element as shown in Figure 3.13.

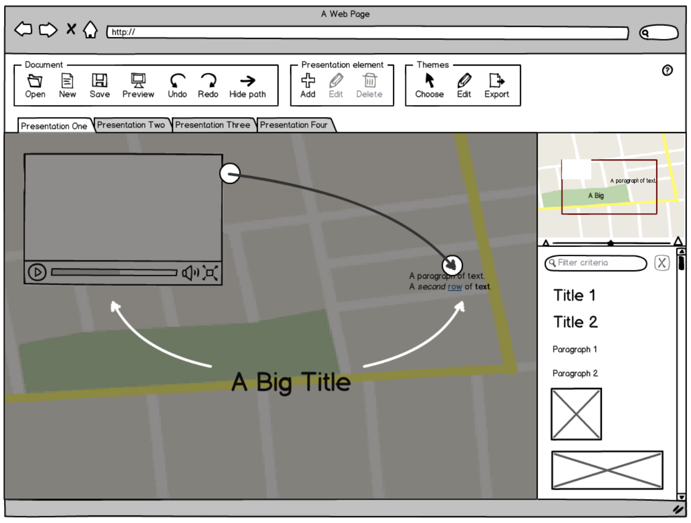


Figure 3.13: The path in a presentation can be edited

It should also be possible for the user to rightclick on a path element to perform certain actions. For example, deleting that specific path element is possible via the appearing context menu as illustrated in Figure 3.14.

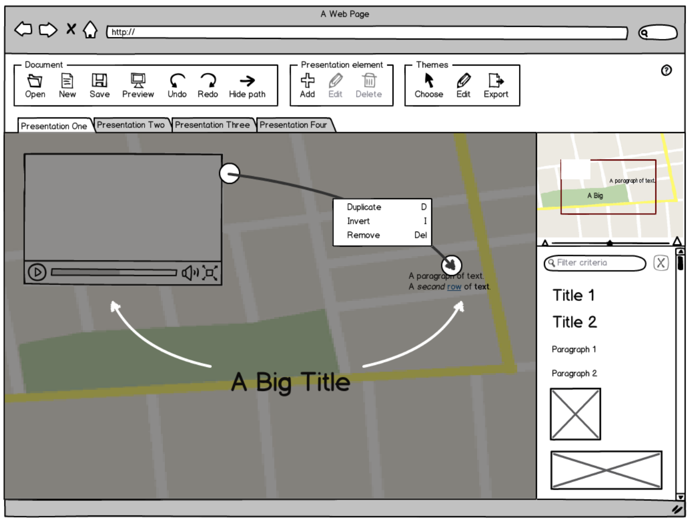


Figure 3.14: Contextmenu that helps the user to edit the path in a presentation

3.5.8 Choosing a theme

Although all presentation tools showed a preview of the themes via thumbnails as we identified in Section 2.1.1.6, there was no live preview available. We propose a system where the live preview of themes is available to the user. For the best user-experience, we propose that the theme selector is displayed over other interface elements as in Figure 3.15 to avoid that the original presentation is covered. This will allow the user to have a more complete overview over their presentation's possible new look. When the user hovers over elements in the theme chooser, the themes are temporarily applied to the presentation. If the user actually clicks on a theme, the theme is definitely applied.

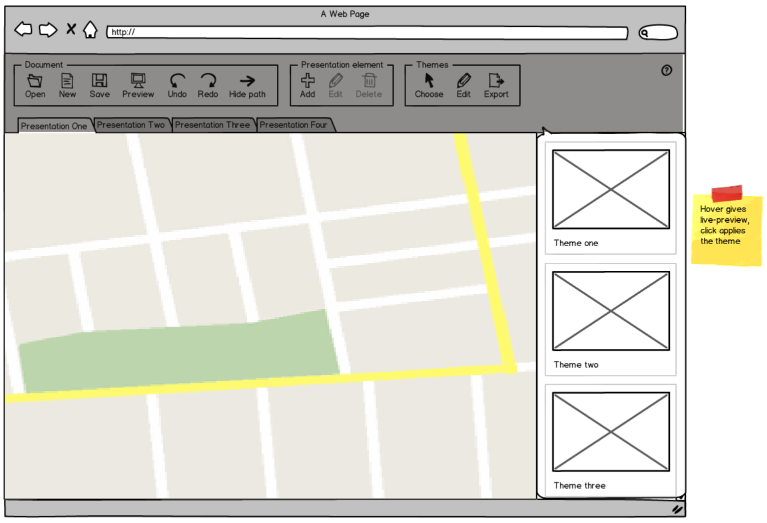


Figure 3.15: Choosing the theme of a presentation

3.5.9 Zooming in a presentation

As we have seen in Section 2.1.1.7, zooming is a tedious task. Only Prezi was able to provide a good zooming mechanism. All the other tools had some kind of sorting mechanism but it was very limited in the different zoom levels and/or it did not provide live previews while changing the zoom level. For us, providing a live preview when zooming is a first requirement to have a good presentation tool. However, zooming can still be massively improved.

3.5.9.1 Semantic zooming

Semantic zooming is a technique where displayed objects reveal more or less details when zooming in or out. When going past a certain threshold, the content displayed may change. In the case of an image with a caption, the caption might be hidden when zoomed out but shown when zooming in.

Experimental evaluations have been performed to see how semantic zooming helped to visualise the source code of computer programs and their internal structure [69]. The results were that the speed of users to perform certain tasks increased. They introduced the concept of continuous semantic zooming which improves semantic zooming by using distortion techniques while zooming. Continuous semantic zooming decreased the amount of time that users needed to perform certain tasks and improved the accuracy.

3.5.9.2 Space-scale diagrams

Space-scale diagrams were introduced as a technique to better understand multiscale interfaces [27]. These types of diagrams have helped in designing good zooming/panning trajectories in Pad++ [10]. Space-scale diagrams help to visualise semantic zooming by showing an object in all its scale-dependent versions at the same time. Thanks to this, designers are able to design better semantically zoomable objects.

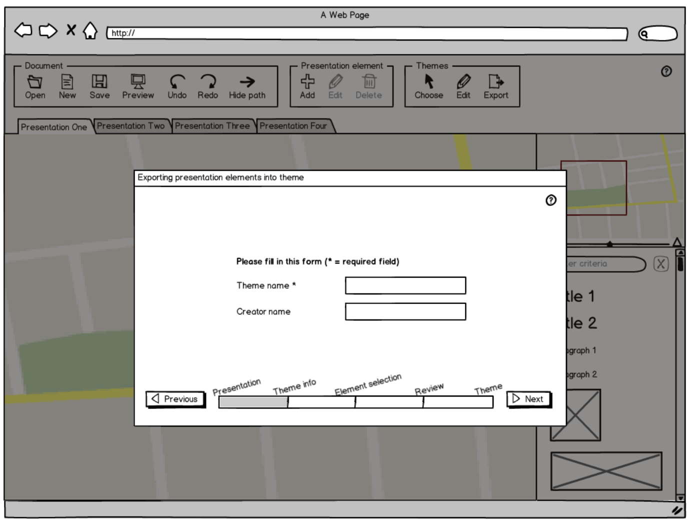
3.5.9.3 Conclusion

It is a rather self-evident choice to use a continuous semantic zooming technique to allow the user to zoom in and out in the presentation editor. The zooming capabilities should be available via natural methods like scrolling with the mouse, preferably with a visual feedback like the actual percentage of zooming. By using space-scale diagrams, good semantically zoomable objects can be designed which will increase the visual perception by the user.

3.5.10 Exporting elements to a theme

In Section 3.4.3, we proposed an additional link between the theme and presentation editors that would allow the creation of new themes based on presentation style definitions. In Section 3.3, we introduced the concept of transclusion which makes exporting elements a bit easier. A new theme can be created by simply referring to the base theme of the presentation while adding the custom style definitions of the presentation.

The problem is that simply adding the custom style definitions will not suffice. An easy example of where this might give unwanted behaviour is when multiple elements of the same type have different custom style definitions. For instance, multiple text elements may have stylings where there are different font sizes applied. When simply adding all these style definitions to the new theme, we actually have conflicting style definitions in our theme and it is not clear which one should be applied to the presentations.


Exporting presentation elements into theme

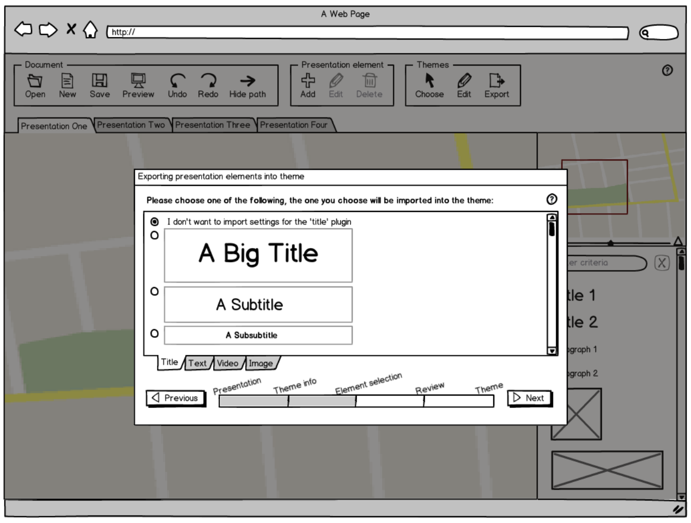
Please fill in this form (* = required field)

Theme name *

Creator name

Figure 3.16: When exporting the stylings of a presentation to a theme, basic info must be given

To solve this problem, we can limit the possibilities to the user when they are exporting the style definitions. We propose to have a wizard in the presentation tool which guides the user through different choices that they have to make. The first screen in the wizard is shown in Figure 3.16. It is simply a form that allows the user to give details about the theme they want to create, like the name of the new theme. The second screen (see Figure 3.17) gives the user an overview of all the different style definitions per type. For example all the style definitions for a text element are grouped and the user has the possibility to select one of them or simply notify that they do not want to add text element style definitions to the new theme. These choices are available for any type of presentation element that is in the current presentation from which they are exporting.


Exporting presentation elements into theme

Please choose one of the following, the one you choose will be imported into the theme:

[✓ I don't want to import settings for the 'title' plugin]

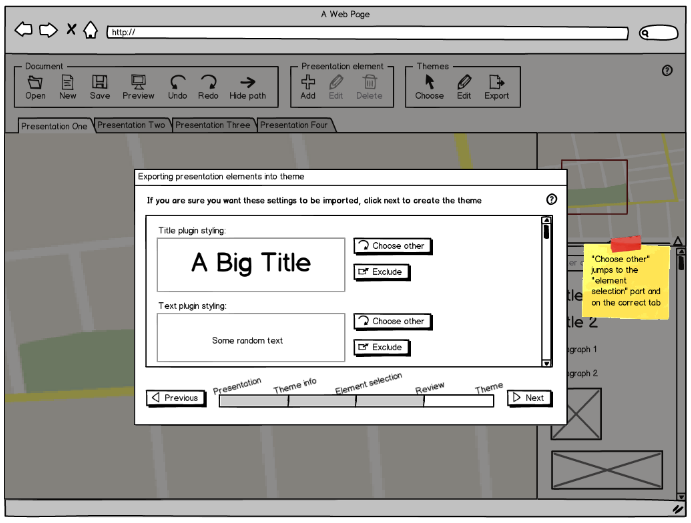
A Big Title

A Subtitle

A Subsubtitle

Figure 3.17: To export stylings of a presentation, the styled elements must be selected

A third and final screen as illustrated in Figure 3.18 allows them to review the choices they have made. In here, a preview of the theme may be shown, as explained later in Section 3.6.6, or an overview of all the selected options. Via this screen, the user can go back to the list of choices in the previous screen or they can create the actual theme.


Exporting presentation elements into theme

If you are sure you want these settings to be imported, click next to create the theme

Title plugin styling:

A Big Title

Text plugin styling:

Some random text

"Choose other" jumps to the "element selection" part and on the correct tab

Figure 3.18: Before the theme is created when exporting, the user can review his selections

3.6 The theme editor

3.6.1 The interface

The interface of the theme editor is in some parts similar to that of the presentation editor. A global menu is also visible and allows full control over the theme editing as shown in Figure 3.19. In contrast to the theme editor, there is no sidebar available.

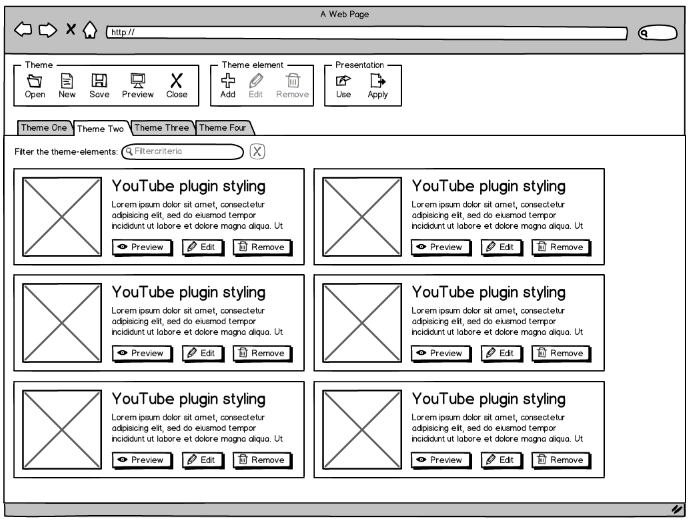


Figure 3.19: The overview of a theme in the theme editor

Under the menu, there are tabs available for the user to switch between the themes that they are currently editing. The user has the possibility to have multiple themes opened at the same time. Whenever all themes are closed, the user is returned to the starting screen as illustrated in Figure 3.5.

Just like in the presentation editor, the menu is horizontal with different groups in it that group certain functionality. The first group is a more global group that allows the user to maintain themes in terms of opening, creating, closing, saving and so on. The second group is more related to the editing of the currently active theme. It allows adding, editing and deleting theme elements. The third and final group consists of the links with the presentation editor. It makes it possible to use a theme in a new presentation and to apply a theme to a set of presentations as we described in Section 3.4.2 and 3.4.4.

The actual content of the theme is shown in boxes that are ordered in a grid-like fashion. Each box represents one element. We propose this approach because a theme is in fact a collection of styling definitions for different types of elements. As each type is represented by a box, the user can get a direct overview of all the elements that are defined in the theme. Via a simple filter option, the field of boxes is filtered based on the keywords that the user provides. This filter is updated live as the user is typing.

Each box should show the type of the theme element, the description that goes along with that type and a predefined thumbnail. This allows the user to better grasp what elements are already defined in the theme.

3.6.2 Creating a theme

Creating a theme is possible via the start screen or in the menu at the top of the application while the user is in the theme editor. To create a theme, only basic information is required as illustrated in Figure 3.20.

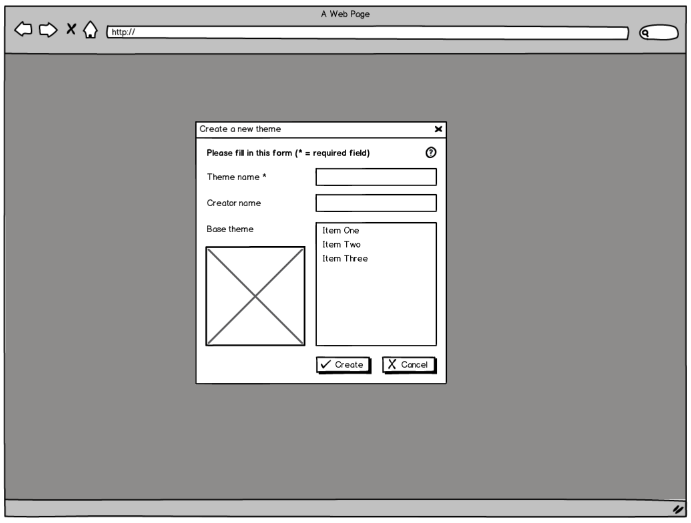


Figure 3.20: Before creating a theme, some basic info must be given

3.6.3 Adding elements

Adding an element to a theme happens via a menu that is very similar to adding elements to a presentation, as highlighted in Figure 3.21. The menu is actually exactly the same as the one in the presentation editor with one big difference. The possible list of options shortens when a theme has more theme definitions. Every time a theme element is added, it is removed from the list of options in the menu. Still, the menu can get pretty big and thus we provide the exact same interface where the user can filter the elements based on keywords that he types. The possible use of fisheye menus [7] should also be taken into consideration.

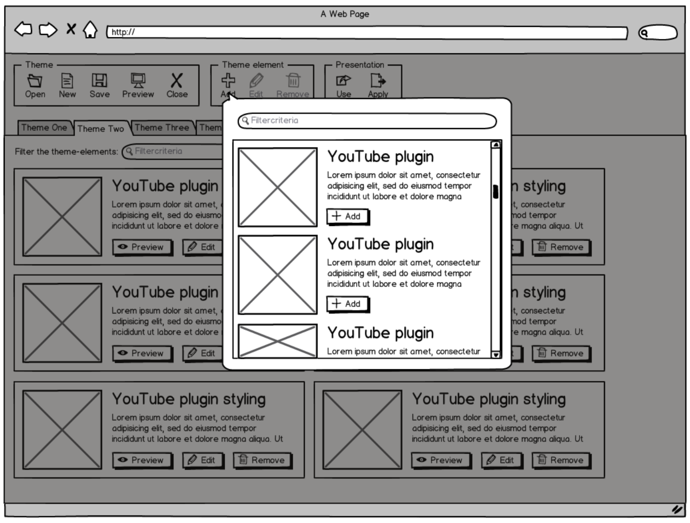
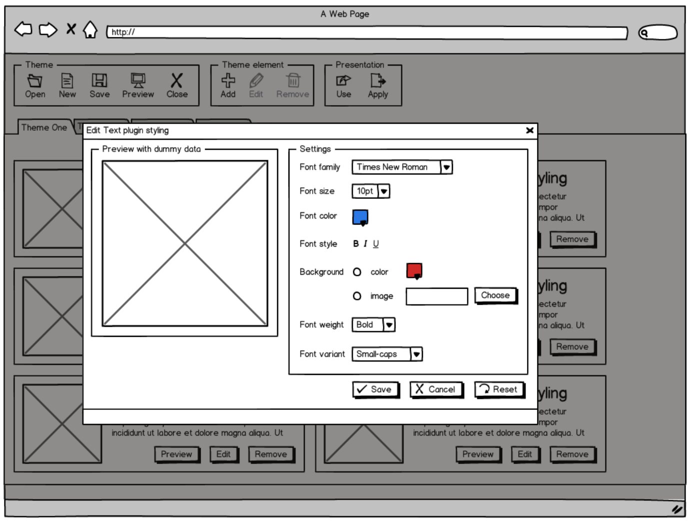


Figure 3.21: Adding new theme elements to define new settings

3.6.4 Changing settings

Every box in the theme editor has buttons that allow the user to start editing the theme definitions. Clicking on *Edit* will trigger a dialogue as can be seen in Figure 3.22. This dialogue is split into a preview part and a settings part.



Font family: Times New Roman
Font size: 10pt
Font color: 🟦
Font style: B I U
Background: ○ color 🟥 ○ image Choose
Font weight: Bold
Font variant: Small-caps

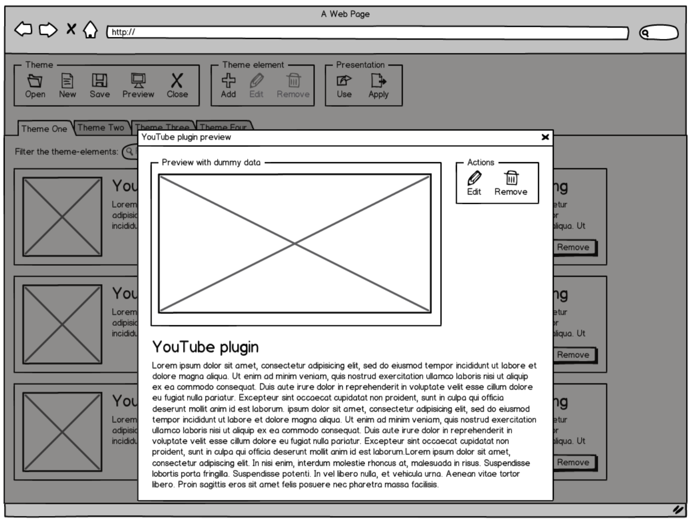
Figure 3.22: Editing the stylings of a theme element

The purpose of the preview part is to give the user a preview of how the definitions affect the resulting output of the theme element. It is updated live whenever a change occurs and allows the user to see the element without possible distracting surroundings.

The settings part is a listing of all the possible definitions for that particular theme element. For example, for a text element the settings can be things like the font family, the font size, the font colour and so on. This list of settings is generated based on the theme element that is currently being edited. Any change in the settings is directly propagated into the preview so the user has direct feedback.

3.6.5 Previewing a theme element

While the user is editing a theme, they can preview the different elements as can be seen in figure 3.23.



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.Lorem ipsum dolor sit amet, consectetur adipiscing elit. In nisi enim, interdum molestie rhoncus at, malesuada in risus. Suspendisse lobortis porta fringilla. Suspendisse potenti. In vel libero nulla, et vehicula urna. Aenean vitae tortor libero. Proin sagittis eros sit amet felis posuere nec pharetra massa facilisis.

Figure 3.23: The user can get a preview of a theme element with extra info

3.6.6 Previewing a theme

When editing a theme, the user might want to get an overall view of how everything looks together. We therefore propose to have the functionality of previewing a theme. A preview is generated based on the theme definitions that are made. Every theme element is in fact rendered with dummy data that is defined in the original theme element definitions. All these rendered theme elements are then put together in a linear presentation structure as shown in Figure 3.24. This allows the user to get an idea of how the theme elements look like when surrounded by other theme elements. We have chosen for a linear structure as we wanted to give the user a very simplistic interface. By providing a linear structure, we allow the user to see the whole preview by simply scrolling down whenever the theme preview might not fit their screen.

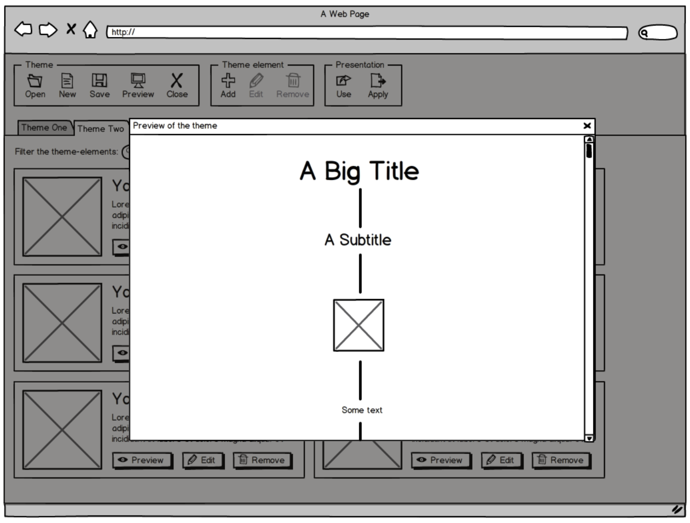
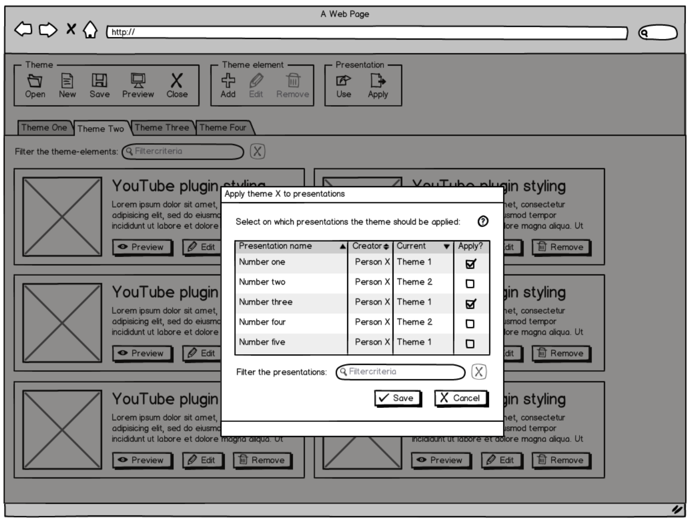


Figure 3.24: A preview of a theme is generated based on the theme definitions

3.6.7 Applying a theme

One of the links between the presentation editor and theme editor is the ability to apply a theme to a presentation without opening that presentation. This comes in handy when a user has for example multiple themes with only the colours differently. As this does not influence the position of the themes and when the colours are carefully chosen, the user does not even have to verify if applying a certain theme does not negatively influence the appearance. We propose a basic interface where the user can have an overview of all the presentations so they can choose on which presentations the theme should be applied as can be seen in Figure 3.25.



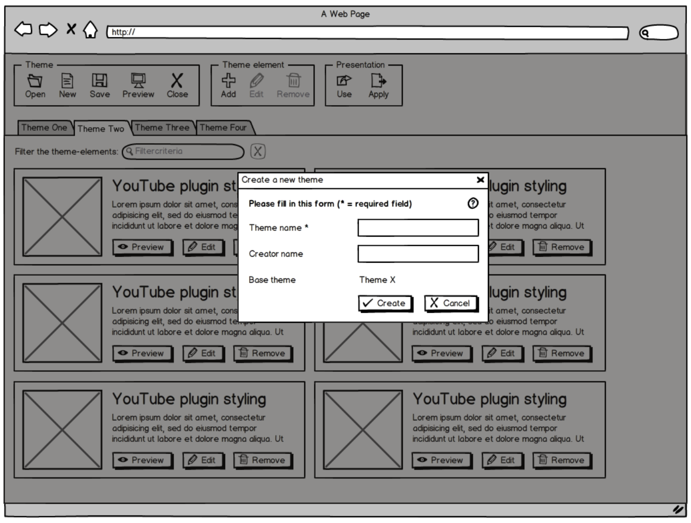
Select on which presentations the theme should be applied:

Presentation name	Creator	Current	Apply?
Number one	Person X	Theme 1	✓
Number two	Person X	Theme 2	☐
Number three	Person X	Theme 1	✓
Number four	Person X	Theme 2	☐
Number five	Person X	Theme 1	☐

Figure 3.25: The user can select the presentation where he wants to apply the theme on

3.6.8 Use a theme

Another link between the presentation and theme editor is the ability to use a theme directly in a new presentation when editing the theme. In current tools, the user is forced to create a presentation in which they can create a new theme in the master slides. But what if they want to work the other way around? What if they would like to create a theme at first and then directly start creating a presentation with that new theme? Because the whole authoring system of themes and presentations is split and we work with transclusion, this becomes possible. We propose to have a basic button or interface element via which the user can start creating a new presentation with the currently opened theme already set as the base theme. The result would be that they get to see almost the same screen as when they creates a new presentation, as illustrated in Figure 3.26.



Create a new theme

Please fill in this form (* = required field)

Theme name *

Creator name

Base theme Theme X

✓ Create X Cancel

Figure 3.26: *The user can directly create a new presentation with the currently opened theme set as the base theme*

3.7 Conclusion

We have presented the different interface elements and interface types that the most ideal presentation tool for authoring MindXpres themes and presentations should have. The presentation tool is divided into two separate editors which are coupled together via four possible interfaces:

- Editing the current theme
- Use a theme in a new presentation
- Exporting stylings to a new theme
- Apply a theme to presentations

Because we separate the tool in a presentation and theme editor, the user has the freedom to edit presentations and themes separately which is not possible in current presentation tools. However, because of the four links that we propose, the user maintains the flexibility to change from editing presentations to themes and back without having the extra hassle of opening a separate program.

Only one of the four links is currently available in presentation tools, the others are either novel concepts or shortcuts. The exporting of theme definitions to a new theme is an interesting approach as it allows the user to save custom stylings from a presentation into a nice and clean theme.

By allowing the user to apply a theme to presentations and by using transclusion of themes into presentations, we also remove one of the biggest problems in current presentation tools: the duplication of data and the fact that presentations are not updated when a theme is changed. With these functionalities, the user can edit or create a theme in the theme editor without having to open the presentation editor. In case of wanting to change the theme of 100 presentations, the user will be able to do this in a couple of seconds where in current presentation tools they would have to edit all 100 presentations by hand.

We explored the possibility of using a zoomable user interface for the authoring of the presentations. By proposing interesting concepts like semantic zooming, the user can get a minimalistic interface that allows them to fully benefit from the zooming capabilities of MindXpres presentations while editing.

The path editor that is currently available in Prezi was already interesting but we extended the capabilities by proposing a very flexible path editor. We propose to have a path editor where the origin and target of each edge is changeable and where there are no limitations on how many edges can depart from a certain presentation element.

The theme editor also introduced novel concepts. We show the theme to the user as a collection of theme definitions for possible theme elements. Via filtering functionality, the user can easily get an overview of what elements are defined while being able to quickly find a specific element that they might want to edit.

We propose a theme preview functionality which is not available in current presentation tools. The preview functionality generates a dummy presentation that gives the user an idea of how it would look like in a basic presentation. The user is thus capable of defining a theme without ever having to test it out on a presentation.

By proposing multiple novel features for a good presentation tool, we believe that our proposed presentation authoring tool introduces enhancements on multiple levels. It is mainly targeted to MindXpres presentations as it relies on a zoomable user interface, the concept of a flexible path in the presentation and so on. Still, some concepts like the exporting of theme element stylings are more general and could enhance the current presentation tools too.

4 Solution

In Chapter 3 we introduced the design for a better presentation tool. In this chapter, we explain the actual implementation that we made. We first start out with a comparison between JavaScript libraries in Section 4.1. The purpose was to determine which JavaScript library would act as a basis for our implementation. In Section 4.2 we designed the general JavaScript architecture that we were going to use. We end the chapter with a some more details of our implementation in Section 4.3 and an evaluation of our tool in Section 4.4.

4.1 Choosing the technologies

4.1.1 Introduction

As the web got bigger and bigger, popularity of JavaScript grew over recent years. We saw the arrival of an enormous amount of JavaScript libraries and frameworks, each with their own purpose or audience. Among these JavaScript libraries and frameworks, several became very popular and jQuery is always considered as the most popular one [1].

We had to make a choice which JavaScript libraries we would take into account for this thesis. Based on previous research [32, 31, 49] and considering possible new contenders, we came up with the following list of JavaScript libraries that we would compare:

- Dojo [2]
- jQuery [3]
- Mootools [4]
- Prototype & Script.aculo.us [5][6]
- YUI [7]

Comparison can happen on several points, ranging from the size of the libraries to the different features that they provide. We have chosen to compare them on the most interesting points that are possibly relevant for building user interfaces in web browsers. This way, this research could be reused in future applications as it can work as a guideline to which JavaScript library is interesting to use.

[1] `https://blog.whitehatsec.com/analysis-of-javascript-library-popularity/`
[2] `http://dojotoolkit.org/`
[3] `http://jquery.com/`
[4] `http://mootools.net/`
[5] `http://prototypejs.org/`
[6] `http://script.aculo.us/`
[7] `http://yuilibrary.com/`

4.1.2 Feature comparisons

4.1.2.1 License

	Dojo	jQuery[i]	MooTools	Prototype[ii]	YUI
Version	1.8.1	1.9.1	1.4.5	1.7	3.7.3
License	BSD or AFL	MIT	MIT	MIT	BSD

[i] We use the extra jQuery UI module [ii] We use the extra script.aculo.us module

Table 4.1: Comparison of JavaScript libraries: Listing the versions and what licenses can be applied

The first comparison point is the license. All libraries provide a quite liberal licensing option, none of them are commercially bound to any company. This is a good thing for every library listed as it gives more freedom to using them.

Conclusion We can conclude that it does not really matter which of the five libraries one chooses when it comes to licencing. All of them have a good license coupled to them.

4.1.2.2 DOM functions and language extension comparison

	Dojo	jQuery[i]	MooTools	Prototype[ii]	YUI
DOM Selection	by id, by class, by attributes, CSS selectors, parent, child, siblings, content, visibility	by id, by class, by attributes, CSS selectors, parent, child, siblings, content, visibility	by id, by class, by attributes, CSS selectors, parent, child, siblings, content	by id, by class, by attributes, CSS selectors, parent, child, siblings, content	by class, CSS selectors
DOM Manipulation	create, append, insert, replace, remove, clone, wrap	create, append, insert, replace, remove, clone, wrap	create, append, insert, replace, remove, clone, wrap	create, append, insert, replace, remove, clone, wrap	create, append, insert, replace, remove, clone, wrap
CSS Func.	get, set	get, set	get, set	get, set	get, set
Events	bind, unbind	bind, unbind, fire, toggle	bind, unbind, fire, toggle, clone	bind, unbind, fire, toggle	bind, unbind, fire, custom event types
Array Functions	iterate, filter, map, some, every	iterate, filter, map, merge	iterate, filter, map, some, every, clone	iterate, compact, clone	iterate, filter, map, some, every
String Functions	pad, repeat, substitute, trim	trim	trim, camelCase, hyphenate, rgbToHex, ...	camelize, capitalize, dasherize, evalScripts, ...	N/A
Math Functions	N/A	N/A	random, limit, times	times, succ	N/A

[i] We use the extra jQuery UI module [ii] We use the extra script.aculo.us module

Table 4.2: Comparison of JavaScript libraries: DOM functions and language extension comparison

DOM Selection By default, JavaScript has methods like `getElementById()` and `getElementByClassName()` which allow the programmer to select elements in the DOM tree. However, these methods are quite limited and require a lot of effort from the programmer if they want to select for example the fifth `p` tag in the `div` tag that has a certain attribute. Libraries provide specialised DOM selectors that allow the programmer to write down complex selectors without having to program difficult structures.

DOM Manipulation Whenever a programmer selects an HTML element, the programmer might want to manipulate it. Inserting a new element, appending a string, setting a value, etcetera are all possible manipulations the programmer wants to perform. The more DOM manipulators the library offers, the more possibilities the programmer has to program in a convenient way.

CSS Functions These are functions that are used to get and set the CSS properties of DOM tree elements. This allows developers to manage for example the proportions of elements, the positioning of elements and so on.

Events When programming a rich web application, one is most of the time programming in an event-driven fashion as one is primarily building the user interface in HTML & CSS. Thus, a developer must be allowed to handle events like mouse clicks on buttons and keyboard events. Some libraries handle events differently so it is appropriate to compare the event handling.

Array Functions In JavaScript, the programmer is able to have collections of data in arrays and objects [19]. As arrays are the primary data structure for having large collections in JavaScript, most JavaScript libraries offer functions to manage arrays better. They allow developers perform operations on arrays like filter, extract, fold and so on.

String Functions As JavaScript is the only language widely supported for programming on the client side [-] and JavaScript is targeted to program dynamic features in the DOM tree, the developer may use strings extensively. If a JavaScript library offers extra string functions, this is a good thing as it allows the developers to write down more abstract code. Here we see that libraries like Prototype and MooTools have extra functions to format for example an URL more (with `dasherize` for example), to do special type of capitalization and so on. We note that YUI has no support for extra string functions and that jQuery only supports a `trim` method.

Math Functions JavaScript only knows one numeric data type: number. With this data type, the use of floating point numbers is possible but basic mathematical functions are very limited. Some libraries provide extra functions that allow easier coding. For example MooTools supports mathematical functions like a `random(min, max)` which gives back a random number between `min` and `max`. Another function is for example `limit(min, max)` which returns a number converted to fit between these bounds. Because these are fairly trivial examples that can easily be reprogrammed, the support of extra mathematical functions besides the standard functions does not weigh much in our comparison.

Conclusion Out of the five JavaScript libraries, MooTools supports the most DOM functions and language extensions. YUI is the only library that is quite limited in its DOM selectors which is a shame as we see that the other libraries offer a very rich selection of DOM selectors: either directly or via handling DOM elements like calling *parent()* on the elements. All libraries support the basics for DOM manipulation, CSS functions and event handling, which is what we would expect from a major JavaScript library. When we compare the language extensions like extra array, string and math functions, we see a bigger variety. YUI is the clear loser because it only adds array functions that almost all the other libraries support too. The clear winner is MooTools which is the one that supports the most array functions, the most math functions and also offers a wide selection of string functions.

4.1.2.3 Visual effects comparison

	Dojo	jQuery[i]	MooTools	Prototype[ii]	YUI
Show/hide	✓	✓	✓	✓	✓
Slide/blind	✓	✓	✓	✓	✓
Resize	✓	✓	✓	✓	✓
Opacity change	✓	✓	✓	✓	✓
Animation	✓	✓	✓	✓	✓
Drag & drop	✓	✓	✓	✓	✓
Effect queue	✓	✓	✗	✓	✗

[i] We use the extra jQuery UI module [ii] We use the extra script.aculo.us module

Table 4.3: Comparison of JavaScript libraries: Visual effects comparison

Show/hide Showing and hiding elements in a webpage can make life easier. It allows the developer to preload elements and hide them so the user has a better user experience as there is no delay for loading in new elements. Although a developer can hide elements himself by editing the *visibility* or *display* attribute in CSS, it comes in handy if the library supports *show* and *hide* functionality. Most of the times, this is linked with possible animation which can affect the user experience in a positive way.

Slide/blind The slide and/or blind functionality is a method to show or hide an element. A slide might be the entering of an element in the screen while a blind is the effect when an element is rolled up like a window shade.

Resize Resizing elements may sound trivial but this is by default not possible for the user in HTML and CSS. Via JavaScript, a developer can set new dimensions to any element which will result in a resizing of the element but the user of the webpage has no control over it. Many JavaScript libraries offer the functionality to enable the resizing of elements by the user of the webpage.

Opacity change By being able to change the opacity, one can animate the showing and hiding of elements for example. Most of the time this is used in the background by animation effects but it also allows the developer to create semi-transparent elements.

Animation Allowing the user to visually track changes that occur in an interface, allow the user to understand more what has happened between the old state and the new state of the screen. These changes can be shown by the use of animations [15]. Animations can be multiple things: moving elements and transitions between elements are a few examples of animations. Studies have shown that decision making is highly dependent on the animation used in user interfaces such as smoothness of transitions, interactivity styles and so on [33].

Drag & Drop Drag-and-drop is an alternative to a point-and-click interaction in user interfaces. Instead of pointing with the mouse and clicking on the target, the user is able to drag elements and drop them on the target. Although research with children has shown that they prefer a point-and-click interaction style instead of a drag-and-drop interaction style [41], the use of a drag-and-drop interaction style may not be a bad choice. As it turns out, there is a difference to the performance index when using different input devices [47]. When using devices like a tablet, the point-and-click interaction style outperforms the drag-and-drop interaction style but when using a mouse, the drag-and-drop interaction style outperforms the point-and-click interaction style. It is therefore reasonable to compare the possibility to enable drag-and-drop functionality on DOM tree elements via JavaScript libraries.

Effect Queue The effect queue is simply the possibility to queue effects after one another. This allows developers to create chained effects that are each activated sequentially. For example, it allows the developer to do a slide, followed by an opacity change, followed by resizing an element. We see that only MooTools and YUI do not support an effect queue while this is possible in the other libraries.

Conclusion We clearly see here that all libraries support everything. The only difference exists in the effect queue which is not available in MooTools and YUI. This could be a possible negative point when choosing a JavaScript library, for example when you want to have multiple effects after each other without having them interfere each other.

4.1.2.4 AJAX and JSON support comparison

	Dojo	jQuery[i]	MooTools	Prototype[ii]	YUI
AJAX	✓	✓	✓	✓	✓
JSON	✓	✓	✓	✓	✓

[i] We use the extra jQuery UI module [ii] We use the extra script.aculo.us module

Table 4.4: Comparison of JavaScript libraries: Support for AJAX and JSON comparison

AJAX Support A couple of years ago, Asynchronous JavaScript And XML has been put forward as a new concept that could help build Rich Internet Applications [58]. It is already been proven that it is beneficial to integrate AJAX functionality into a web application. It is beneficial on the amount of time an user needs to complete a task or on the general satisfaction they feel [43]. Although not all rich web applications need AJAX, if a library supports AJAX it could help to raise the level of usability and satisfiability in a web application. It allows a developer to move certain functionality to a server and let the web application interact with the server without having the need to refresh the web page.

JSON Support The JavaScript Object Notation [1] is a lightweight data interchange format that claims to be easier to read for humans and computers. It is a subset of the JavaScript standard [2] that works well as a data interchange language. Many languages support it so it comes in handy if a JavaScript library supports it too. This allows developers to transmit data back and forth between the client and server without having to know what languages are used on the server.

Conclusion All libraries support AJAX and JSON which is a good thing. This allows a developer to put certain functionality on a server. The client can communicate with the server via AJAX which is non-blocking and all data can be put in a JSON format for better interoperability so it does not matter which languages one uses on the server, as long as they can parse JSON.

[1]http://www.json.org/

[2]JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999 http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf

4.1.2.5 Widgets comparison

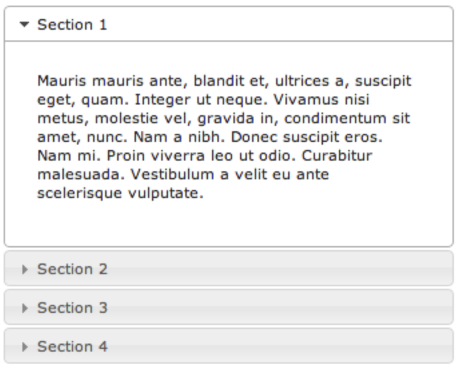
	Dojo	jQuery[i]	MooTools	Prototype[ii]	YUI
Accordion	✓	✓	✓	✗	✗
Autocomplete	✓	✓	✗	✓	✓
Colour picker	✓	✗	✗	✗	✓
Date picker	✓	✓	✗	✗	✓
Dialogues	✓	✓	✗	✗	✓
List sorting	✗	✓	✓	✓	✓
Menus	✓	✓	✗	✗	✓
Progress bar	✓	✓	✗	✗	✓
Selection list	✗	✓	✗	✗	✗
Table sorting	✓	✗	✓	✓	✓
Tabs	✓	✓	✓	✓	✓
Tooltips	✓	✓	✓	✗	✓
User editing	✓	✗	✗	✗	✗
WYSIWYG editor	✓	✗	✗	✗	✓

[i] We use the extra jQuery UI module [ii] We use the extra script.aculo.us module

Table 4.5: Comparison of JavaScript libraries: Widgets comparison

Accordion An accordion is an user interface element which has visual similarities to the music instrument. The idea is that one can have a list where the elements collapse and expand when the user clicks on them or hovers over them with the cursor.

Figure 4.1: Example of an accordion user interface element where sections are collapsed and expanded when the user clicks on them



Mauris mauris ante, blandit et, ultrices a, suscipit eget, quam. Integer ut neque. Vivamus nisi metus, molestie vel, gravida in, condimentum sit amet, nunc. Nam a nibh. Donec suscipit eros. Nam mi. Proin viverra leo ut odio. Curabitur malesuada. Vestibulum a velit eu ante scelerisque vulputate.

Autocomplete Autocomplete is a user interface concept where suggestions are shown as the user is typing in a text box. This is an extra help for the user as they see suggestions of what he is actually trying to type or search [52].

Colour picker A colour picker is in most cases a hovering user interface element that allows the user to select a colour in a colour palette. This is more convenient than asking the user to type in for example the RGB values manually. Studies have shown that high visual feedback increases the accuracy of picking the desired colour [21].

Date picker A date picker is in most cases a hovering user interface element that allows the user to select a date in a calendar.

Dialogues A dialogue is an extra screen that is used to enable reciprocal communication between a computer and its user. In other words, it allows the user to have a *dialogue* with the computer by confirming for example the deletion of an element.

List sorting List sorting is a functionality that allows the user to sort the elements in a list without the developer having to program the entire sorting algorithm.

Menus Menus are an essential user interface element as they allow the grouping of different functionalities in a convenient structure.

Progress bar A progress bar is a bar in the interface that gets filled as the user is progressing. This gives back feedback to the user of how far he is in the process of completing a task.

Selectable elements HTML5 supports the select tag which allows one to select elements in a list. In an user interface, it might happen that a developer does not want a list to be selectable but for example a grid. The HTML5 select tag is thus too limited and several libraries support the possibility to allow other elements to be selectable.

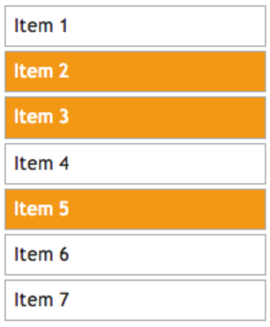


Figure 4.2: Multiple DOM tree elements can be made selectable in jQuery

Table sorting Table sorting is a functionality that allows the user to sort the rows in a table without the developer having to program the entire sorting algorithm.

Tabs Tabs are used for example when showing the user that they have multiple documents opened.

Tooltips Tooltips are small hovering elements that appear when the user for example hovers over an image to get more information about what is shown on the image.

User editing User editing is the enabling of editable elements in the web page. By default, HTML elements are not editable except for `input` and `textarea` elements. By having the functionality to make other elements editable, it is for example possible to make a paragraph of text editable.

WYSIWYG editor A What You See Is What You Get editor is helpful for the user for example when editing posts in a blog. It gives direct feedback of what will be shown on the final page.

Conclusion It is hard to draw conclusions from the widgets comparison because this is very application dependent. For example if an application does not need native support for dialogues, MooTools is a good choice but if one will use dialogues extensively it may not be the best choice. There are also a lot of plugins available for most libraries which can allow one to add certain widgets, we only have compared the ones that are natively supported in the libraries. Purely based on the amount of supported widgets, Dojo is the best library.

4.1.3 Usability testing

In the previous section, we have compared five JavaScript libraries and have seen that they are fairly competitive to each other. But how good is their usability? How can we test the usability? We set up a small test in which we gradually try to find problems as we implement more features. After each implementation, we evaluate how well it went and what possible problems were.

4.1.3.1 Hello world!

We started with the easiest part: setting up a webpage for each library in which we load the library and let it print ‘Hello world!’, a classic hello world application. In plain JavaScript a hello world application which puts ‘Hello world!’ in the body of the webpage would look like this:

```
document.write('Hello world!');
```

Listing 4.1: A hello world application in plain JavaScript

This code is simple and clean, it can be put in the HTML itself but it can also be put in a separate JavaScript file without a problem. It simply works. We will now see what happens when we try to achieve this when using each library to print ‘Hello world!’. Note that we will not use document.write but the functions the library provides to print text in the body of a webpage.

Dojo Dojo is the first library that we try out. Firstly, we load in the dojo.js file that is delivered in the toolkit and after that, we put this code in a JavaScript file:

```
dojo.ready(function(){
  dojo.query("body").addContent("Hello world!");
});
```

Listing 4.2: A hello world application in Dojo JavaScript library

Here we see the complexity of a JavaScript library already. First, we had to call `dojo.ready()` which takes a function as a parameter. This function waits for the DOM to be completely loaded after which it is executed. In our function, we call `dojo.query()` which searches for the element in the DOM tree we want to manipulate. In our hello world application this is the body element on which we call `addContent()` to add the correct text.

The code is simple and clean, the overhead is minimal. The library was easy to load and worked without a problem.

jQuery The next library that we try out is jQuery, at the moment of writing the most popular library out there [1].

```
$(document).ready(function(){
  $("body").html("Hello world!");
});
```

Listing 4.3: A hello-world-application in jQuery JavaScript library

Just like Dojo, jQuery uses its own variable `jQuery`. However, most people know jQuery for the `$` symbol. jQuery reveals a bit more of what happens under the hood. By using `$(document)` we see that we work with the `document` variable that is available in plain JavaScript as a gateway to the DOM tree. By calling `ready()` we

[1] `https://blog.whitehatsec.com/analysis-of-javascript-library-popularity/`

bind a function to the event 'DOM is loaded and ready'. In our small function we query for the body-element and set the correct HTML to print 'Hello world!'. The `$` can be directly called with a query where in Dojo you had to call explicitly `dojo.query()`. This makes jQuery code a bit shorter but not necessarily easier to read.

MooTools The third library we try out is MooTools. In the past, it also used the `$` variable like jQuery but now it is equipped with a 'dollar safe mode' as they call it. Whenever the `$` variable is already in use, it does not change the contents of `$`. For convenience, we will always use the original function `document.id()` instead of `$`.

```
window.addEvent('domready', function(){
  document.id(document).getElement("body").set('text', 'Hello world!'
      );
});
```

Listing 4.4: A hello world application in MooTools JavaScript library

As we can clearly see, the MooTools code is longer and more explicit. It has no abstractions to add our function to the 'domready' event and setting the HTML to 'Hello world!' is also longer. Via `document.id`, we get the DOM tree itself on which we can query for the body-tag via `getElement()`. This will return an Element object on which we can call `set()` to actually put 'Hello world!' into the webpage. It is not as short as other libraries but it gives the developer more insight in what actually happens.

Prototype The fourth library we try out is Prototype. Just like MooTools and jQuery, it likes to populate the `$` variable. This is shorthand for the `document.getElementById()` method provided by JavaScript and when using multiple libraries, it is good practice to not rely on the `$` variable.

```
Event.observe(window, 'load', function() {
  $(document.body).insert("Hello world!");
});
```

Listing 4.5: A hello-world-application in Prototype JavaScript library

The code is similar to all previous libraries, it waits for the DOM to be loaded completely to execute the given function. Because the `$` is shorthand for `document.getElementById()`, it is counter-intuitive to use `$(document.body)`.

YUI The final library that we try out is the Yahoo! User Interface library. It uses a different loading principle than all the other libraries. Where you had to explicitly say in the other libraries what should be executed as the DOM is loaded, YUI automatically handles this itself. It loads in all the modules that you want which are the first parameters of the `use()` function and after that it executes the function. In this function, the developer can use the Y variable to access the methods from the `node` module, making the code very robust and able to work with other JavaScript libraries as it only knows one global function: `YUI()`. There are no global variables like `$`.

```
YUI().use('node', function (Y) {
  Y.one('body').setHTML('Hello world!');
});
```

Listing 4.6: A hello world application in YUI JavaScript library

Conclusion In all libraries it was fairly easy to create a hello-world-application. All but YUI use the same principle where developers have to explicitly define what should be executed as soon as the DOM is loaded. YUI handles this completely for the developers. All have their own way of defining your own scripts and all of the hello-world-applications could be written down in short amounts of code.

4.1.3.2 Combining functionalities

When using a JavaScript library in a Rich Internet Application, developers do not only use one functionality but combine a lot of them to achieve what they want. For example they want to allow the user to move elements on the screen but also want to add animation, make elements resizeable and so on.

It therefore makes sense to see how well functionalities can be combined when using a JavaScript library. We have seen in Table 4.3 that a lot of visual effects are supported by all the JavaScript libraries. Only the effect queue was not supported by MooTools and YUI. If all these libraries support resizeable elements, draggable elements and so on, does this mean that they can be combined?

We created a basic test case: a webpage with three elements which can be seen in Figure 4.3. The first element should be draggable, the second element should be resizeable and the third element should be draggable and resizeable. With this we test how easy it is to get an HTML element resizeable or draggable and if these two basic functionalities can be combined.

Dojo The first library that we used was Dojo. To get elements to be resizeable and draggable, was very troublesome and we could not end up with a working example. In Dojo, to have a draggable and resizeable element, the developer has to turn his DOM element into a widget. This happens via the *FloatingPane* functionality. With this, the developer can define which element should be turned into a widget but it has its limitations. For starters, one is obligated to define an ID for the DOM element. The second problem is the fact that Dojo turns the element into a widget. A widget in Dojo always has a title and starts hovering on the page unless you can position it via

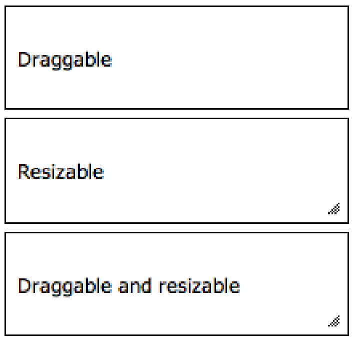


Figure 4.3: The test case that we want to implement to test the combination of functionality

the *position: absolute* attribute back into the HTML. In a dynamic page this is very cumbersome. Overall, our code looked like this:

```
var pFloatingPane;
dojo.require("dojox.layout.FloatingPane");
dojo.ready(function(){
    pFloatingPane = new dojox.layout.FloatingPane({
        title: "A floating pane",
        resizable: true, dockable: true,
        style: "position:absolute;top:0;left:0;width:100px;height:100
            px;",
        id: "pFloatingPane"
    }, dojo.byId("resize"));
    pFloatingPane.startup();
    pFloatingPane.show();
});
```

Listing 4.7: Resizeable and moveable elements in the Dojo JavaScript library

The result is that we could only get one element working. We could not find the possibility to make an element only resizeable without having it turned into a widget or a draggable element. The result of this experiment is shown in Figure 4.4.

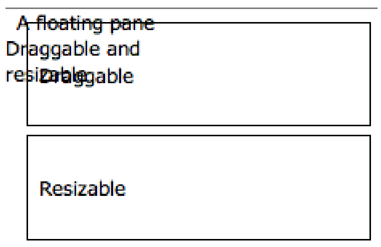


Figure 4.4: The resulting web page when Dojo has added its functionalities

We can clearly see that the layout got messed up: elements started hovering over each other and the styling disappeared thanks to the initialization of the widget which completely overrides all previous styling. To overcome this styling problem, one should be able to completely retrieve all current styling and reapply it. This is clearly not good for maintaining the code and adds a lot of complexity. The result is that Dojo is not usable to make elements resizeable and/or draggable.

jQuery The second library that we used was jQuery. Thanks to the support of functions like *draggable()* and *resizable()*, elements can easily be made draggable and resizeable. When testing the result, we see that the functionalities do not conflict at all. Elements are draggable by clicking on them and moving the cursor around while still pressing the mouse down. When clicking on the edges of the elements, the element is resized without any problem.

```
$(document).ready(function() {
    $(".draggable").draggable();
    $(".resize").resizable();
});
```

Listing 4.8: Resizeable and moveable elements in the jQuery JavaScript library

MooTools The third library that we used was MooTools. In contrast to Dojo, it was much easier to find the correct methods of how we could make elements resizeable and draggable. The resulting code is very short like it was in jQuery.

```
window.addEvent('domready', function(){
    new Drag.Move(document.id(document).getElement(".draggable"));
    document.id(document).getElement(".resize").makeResizable();
});
```

Listing 4.9: Resizeable and moveable elements in the MooTools JavaScript library

However, several problems arise when we look at the result that browsers rendered. The first problem is that the draggable elements jump out of their context. Just like in Dojo, they started to hover the other DOM elements. A second problem arose with the resizeable elements. By default, MooTools does not show a handle on the resizeable elements which indicates that they are resizeable. Simply clicking in the element and moving the cursor around will resize the element. A developer has the possibility to define his own handle but it has to be added before the elements are made resizeable. There is no dynamic insertion of a handle as it would be done in jQuery. Some developers may prefer this, some may not. The final problem is that resize and drag functionalities are not combinable. When adding both functionalities to an element together, the functionalities were not visible and did not work at all. The result is that MooTools is not usable when one needs resizeable and draggable elements, it is only usable for a developer when he only needs one of the two functionalities on DOM elements.

Prototype The fourth library we tried was Prototype with Scriptaculous. In Table 4.3 it says that elements can be made draggable and resizeable. This was based on the documentation that we found. However, for some unknown reason the resizeable functionality disappeared from the Scriptaculous library although we found links and webpages that point to the official Scriptaculous website with the documentation for the resizeable functionality [1]. According to this documentation, the code would look like this:

```
Event.observe(window, 'load', function() {
    $$('div.draggable').each(function(el) { new Draggable(el, {}); })
        ;
    $$('div.resize').each(function(el) { new Resizable(el, {}); });
});
```

Listing 4.10: Resizeable and moveable elements in the Prototype JavaScript library

But since we can not get the Resizeable functionality working (browsers complain about the lack of the Resizable function), the test case fails and we can conclude that Prototype and Scriptaculous are no good candidates. If developers want elements to be draggable and resizeable, they will have to write the resizeable functionality themselves.

YUI The last library that we try out is the Yahoo! User Interface library. The default resize and drag functionality in YUI did not deliver the result we expected. The handlebars for resizing the elements were shown at the corners of the browser's screen. By tweaking the settings, we could achieve the wanted result.

```
YUI().use('node', 'dd-drag', 'resize', 'resize-plugin', function (Y)
   {
    Y.all('.resize').each(function(selectedNode) {
        new Y.Resize({ node: selectedNode,
            preserveRatio: true, wrap: true,
            maxHeight: 200, maxWidth: 400,
            handles: 't, tr, r, br, b, bl, l, tl'
        });
    });
    Y.all('.draggable').each(function(selectedNode) {
        new Y.DD.Drag({ node: selectedNode });
    });
});
```

Listing 4.11: Resizeable and moveable elements in the YUI JavaScript library

The problem however, is there is some very strange behaviour when using this code. The first render of the page looks correct, all handlebars are in place and resizing the elements is no problem. However, the problem arises when we try moving an element that is resizeable (in our set-up the third element). The element itself is moved but its resize handlebars don't move along, they keep hanging on their original position. The result can be seen in Figure 4.5. We can conclude that these functionalities will not work together.

[1] http://script.aculo.us/docs/Resizable.html

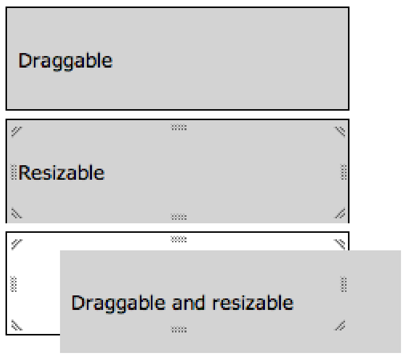


Figure 4.5: The result of moving a resizeable element when using YUI

Conclusion The conclusion of the usability testing is that although all libraries promised a lot of interesting features, almost all of them fail to combine the promised features. When setting up a basic test scenario where we wanted three elements on a page with combined functionalities, we discovered several problems.

The first element on the page had to be draggable. This was no problem for all the JavaScript libraries except with Dojo we ran into problems. The reason behind this, is that Dojo wanted to turn the element into a widget. Although this does not mean that there is automatically a problem when using Dojo, it sadly became a problem in our test scenario. Turning the DOM element into a widget was a very destructive operation. The result was that all the styling of the DOM element was lost thus giving a lot of problems to set everything back to the old styling. The other libraries had no very big problems but jQuery and Prototype handled it the best.

The second element on the page had to resizeable. The biggest notable problem was found in Prototype. Although Scriptaculous promised to have resizeable elements functionality and we found some documentation about it, the result was that there was no resize functionality to be found in their code. Another notable problem could be found in YUI. Here the handlebars to resize elements acted not as expected. When moving an element, one expects the handlebars to move with it which was not the case at all. The only library where making elements resizeable without having the need to add extra HTML yourself, was jQuery. MooTools also did a good job but here you had to add your own HTML for the handlebars.

The third and final element on the page had to be draggable and resizeable. The only library that was able to do this without messing up, was jQuery. YUI almost got it right and in other libraries it was either impossible or everything got screwed up.

We can therefore conclude that jQuery is the best library out of the five we compared when it comes to combining functionality on the same element. Although this may not be a requirement in every programming project, it can give a problem in the future as it shows that not all JavaScript libraries are very robust. Either certain functionality is very cumbersome to add or the functionalities conflict with each other. In the ideal situation, one wants to have all functionalities working together without conflicting.

4.1.4 Conclusion

We did a comparison of five JavaScript libraries on three levels. Firstly, we did a feature comparison after we selected the five JavaScript libraries that we were going to compare. Based on this, we can conclude that all of them are very similar and competitive. Either they all supported the same features or they had certain very competitive features.

The second level on which we did a comparison, was when we constructed a hello world application. This allowed us to analyse some basic behaviour:

- How should the library be loaded?
- How does the library offer its functionalities: does a developer have to load in certain modules or is everything possible from the start?
- How can a developer select the body of a webpage?
- How can a developer set the content of the webpage?

This was very basic behaviour and showed use that all libraries are used in roughly the same way and that it was possible to write a hello world application fairly easily.

The third level on which we did a comparison was in a situation where we wanted to use two main functionalities that are important when creating a user interface which is basically a WYSIWYG: resizeable elements and draggable elements. In this situation, we had troubles either setting up certain functionality or combining the functionalities. Only jQuery could provide us with what we wanted in very short and easy code. All other libraries had problems. Based on this, we can formulate certain conclusions:

- Dojo likes to turn everything into a widget which is a nice conceptual abstraction but can give serious problems. The first problem is that functionalities from separate widgets cannot be combined on the same element. The second problem is that turning a DOM element into a widget is a destructive operation on the CSS styling.
- YUI is very bad at combining certain functionality. Although the functionality on its own works, it can give strange results.
- MooTools does not enable certain functionality when it is combined with other functionality.
- Protoype relies on Scriptaculous to give many visual features which are not always implemented although the documentation about it exists.
- jQuery allows very short code and was the only one that did what we expected.

When we combine these conclusions with the results of previous research [32, 31, 49] where there was testing on the code quality of the libraries, the amount of documentation and so on, we can conclude that jQuery is the best tool for a web browser based user interface.

4.2 JavaScript architecture

In Section 4.1, we decided to use jQuery as the basic library for our implementation. As jQuery is a JavaScript library, we have the complete freedom in how we are going to use it. If it were a framework, a certain software architecture might have been forced like Model-View-Controller (MVC) [44].

Most websites use a JavaScript library like jQuery as an extra layer in the user interface that results in higher usability. For example, direct validation of email addresses, the use of a date selector for date input fields, basic form validation and so on. In such cases, there is no real need for a big software architecture. In most cases, some extra included JavaScript files in the HTML that contain a bunch of methods are sufficient. These methods simply subscribe to certain events like key presses in a certain input field. This method of using JavaScript is okay when the developers are adding extra usability.

In the case of MindXpres, this is not the way to go as we wanted to create a real Rich Internet Application (RIA) [66]. The idea was that it is for the user just like he would open any application on their computer, with the only exception that it runs in their browser.

Although the actual implementation would be some sort of website, we took it a step further by deciding that the application would run completely client-side. In most RIAs, AJAX is used for achieving data consistency and backups while browsing [58]. We decided however to have a single-page web application that is not dependent on a server-side component. A huge advantage of this is that data cannot get corrupt as the browser's *back* button is useless.

The starting point of the web application would simply be an HTML file which renders the default screen and loads in the whole application that is written in JavaScript. After that, the application is completely responsible for removing and adding HTML just like is the case in some AJAX-based web applications.

4.2.1 Organizing the code

In essence, building a web application is writing bits of functionality that have to react to events triggered by the user e.g. clicking on a button, moving the mouse and so on. Some functionality may be complementary while other may work independently. It is even possible that some functionality is not required for a correct working of the application. For example, we could add an auto save feature which saves the state of the application every five minutes. This feature works completely independent of other features and the user is able to work without that specific feature.

4.2.1.1 Modules

After intensively looking around on the Web, we stumbled upon RequireJS[1]. This is a library that helps developers to dynamically load JavaScript code. The code is organised in modules. We decided to use this concept of modules for multiple reasons.

The first one is of course the dynamic loading of code. It would allow us to save a lot of memory if certain functionality would be loaded only when it is necessary. Although computers are always getting more powerful, JavaScript has the limitation that there is no memory management possible. By loading in certain pieces of code only when it is used, we can at least avoid unnecessary use of memory by avoiding unused functionality when starting the application.

The second one is the separation of functionality. When using modules, we can group certain functionality that belongs together. For example functionality related to the editing of a presentation element could be put into one module. This gives a better overview of the code as there is an organisation at the module level.

The third and final one is the reuse of functionality. It is possible to reuse modules like a form validation module that validates for example the formatting of email addresses.

4.2.1.2 File structure

Putting code into a module can be done in multiple ways. One can simply create one huge file with several modules in it but there can also be a certain structure when using one module per file. In the application that we developed, we used the convention that each module has their own unique file. Each module belongs to a certain category to group everything in a logical way. Some examples of these categories are: *presentation*, *presentationelement* and *theme*. We have a `modules` folder where each module is organised as follows: `category/modulename.js`.

Alongside the modules folder, there are also other folders: lib, plugins and templates. The lib folder is short for *libraries*. In this folder, all external libraries like jQuery, jQueryUI, RequireJS and so on are put. The reason behind this is that we wanted to have a separation between used libraries and code that is specific for our application.

The `plugins` folder contains all the plugins that are used in the application. The explanation why we use plugins and how they work, can be found in Section 4.2.2. The last folder is `templates`. In this folder, all the templates are put that are used for the presentation. These are XML files that will be parsed: HTML forms, formatting for dialogues and so on. The reason behind this is very simple: it allows for a clean separation of the presentation and the actual functionality. As a bonus, certain HTML can be reused. A basic string replacer is used as a template parser to insert certain values into the HTML. An example template is given in Listing 4.12. Variables are surrounded with double curly brackets.

[1]`http://requirejs.org/`

```
<li class="presentationtab" data-presentationid="{{id}}">
    <span>{{name}}</span>
    <button class="close edit-presentation-tab-close" />
</li>
```

Listing 4.12: Example template

4.2.2 Plugins

Plugins are an essential part of the MindXpres application and one of the key features. Each element in a presentation is in fact an instance of a plugin. Because all the elements are plugin-based, new types of elements can easily be added.

4.2.2.1 Loading in plugins

The theoretical part of a plugin-system is very trivial but the implementation gave some challenges. For example, JavaScript is not capable of scanning a directory. Because of this, we are not able to let the application scan a directory with all the plugins to load them in automatically when the application is loaded.

We solved this problem easily by adding a JSON-file called activePlugins.json in the directory of plugins in which a developer must register his plugin. The JSON-file with the three plugins we implemented can be seen in Listing 4.13. This file is parsed and we give the assignment to RequireJS to load in all the plugins.

```
{
  "text": "textPlugin",
  "title": "titlePlugin",
  "youtube": "youtubePlugin"
}
```

Listing 4.13: The JSON file listing all the plugins to load

4.2.2.2 Structure of a plugin

In Figure 4.6, we give the general structure of a plugin in our application. Every plugin has four basic properties: name, type, description and settings. The type is a unique identifier that allows instances like a presentation element to be typed. The name and description are merely there to help the user of the application as they are used in the interface when the user wants to add elements to a presentation or theme.

The settings property contains all the elements of a plugin instance that may influence it appearance: text colour, font size, font family, a YouTube URL, width, height and so on. Such a setting is called a plugin setting. For more information on how a plugin setting is actually implemented, we refer to Section 4.2.2.4.

Each plugin has multiple methods: `init`, `create`, `render`, `preview` and `toString`. The `init` method is called when the application is loaded. This can be used to make sure the plugin is ready to be used. For example an external library can be loaded. The `create` method is called when a new instance is requested in the theme or presentation editor. `Render` and `preview` are used to parse a plugin instance to something that can be shown to the user of the application. The `toString` method is an extra render method that returns a string representation of the plugin instance.

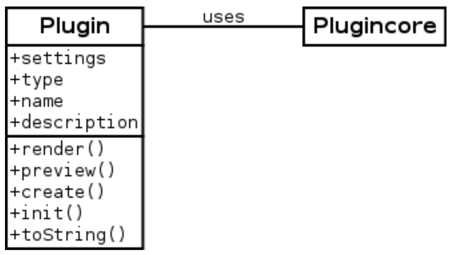


Figure 4.6: UML diagram showing the basic plugin structure

Creating a plugin is easy. One simply has to create a new directory for the plugin with the name of the directory the same as the plugin. In this directory, one has to create a `.js` file, also with the same name as the plugin. In this `.js` file, the developer simply adds a module just like in the rest of the application. Another file that can be created in the directory is a style.css for the custom styling that belongs to that plugin.

The newly created plugin can then be added to the `plugins` directory and by adding an extra line in the `activePlugins.json` file. From then on, whenever the application is started, the plugin will also be loaded.

4.2.2.3 Plugin core

Because all plugins have certain things in common, we created a plugincore which is a sort of base module that all plugins should use. The current structure of the plugin core is shown in Figure 4.7. It offers certain basic methods like create that all plugins support. The idea behind the create method in the plugin core, is that plugins call this method to construct an instance. This is to ensure that all plugins get the exact same interface for the developers and that there is an extra check on the validity of plugins. If a plugin does not offer basic methods like render, they can not be used correctly in the application and might result in big error messages. The use of the plugin core to validate and create plugin instances, is a safe way to guarantee correctness.

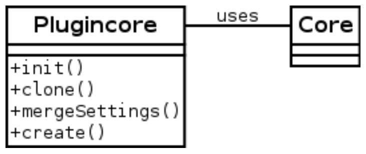


Figure 4.7: UML diagram showing the plugin core module

4.2.2.4 Plugin settings

A plugin can define in its `settings` variable what the different type of settings are that influence the rendered result of a plugin instance. For example for the YouTube plugin, we want to allow the user to add: a YouTube URL, a width and a height.

The first idea to allow such settings was by allowing the use of a markup language in a plugin. The plugin itself would then be programmed to parse the input and the render the correct result to be added in a presentation. There are several problems with this approach.

One of the problems is that it gives a lot more work to plugin developers as they not only have to make sure a plugin instance is correctly rendered but that the settings are correctly parsed.

A second problem is consistency in the formatting of the settings. This is more a usability problem as each plugin may implement text input differently.

The third problem was identified in Section 2.3. Almost no user interface language that allows an abstraction and separation of concerns, is supported by the browsers or there is no renderer available.

To circumvent these problems, we introduced the concept of plugin settings. These are in essence the building blocks of a plugin. It is an abstraction of certain settings that may occur in multiple plugins. A plugin setting can be a line of text, a URL, a colour, a number within a range and so on. Each plugin setting is rendered in its own way. How they are rendered, is defined in the plugin setting itself to create a consistency and an easy-to-use interface. The only thing that plugin developers can do is define what plugin settings are used and read out the values of each element. These values can then be used to process and render a correct version of the plugin instance.

We implemented several plugin settings which plugin developers can use. As can be seen in Listing 4.14, developers only have to list up all the possible settings and what their type is. The one we show is for the text plugin which allows a user to enter text, give it a certain colour and a certain font size which is limited. The application will then parse all these settings and based on the types, it will render the correct input fields. For example for a line of text, an HTML input field will be generated.

```
var settings = {
  textContent: {
    type: 'text',
    id: 'textContent',
    name: 'Text to display',
    value: 'No text defined'
  },
  textColor: {
    type: 'color',
    id: 'textColor',
    name: 'Font color',
    value: '#CCCCCC'
  },
  textFontsize: {
    type: 'int',
    id: 'textFontsize',
    name: 'Font size',
    min: 8,
    max: 20,
    value: 12
  }
}
```

Listing 4.14: The plugin settings for a piece of text

4.2.3 Entry point into the application

```
requirejs.config({
  enforceDefine: false,
  baseUrl: 'src/modules/',
  paths: {
    jquery: '../../lib/jquery-1.9.0',
    jqueryext: '../../lib/jquery-ext',
    jqueryui: '../../lib/jquery-ui-1.10.0.custom.min',
    // ... extra libraries here ...
  }
});
define(
  ['core', 'jquery'],
  function(core, $) {
    require(['plugincore', 'jqueryext', 'jqueryui', '
        jquerycontextmenu','jqueryplumb', 'xslt', 'colorpicker'],
        function() {
      $(document).ready(function() { core.init(); });
    });
  }
);
```

Listing 4.15: Stripped down version of the main file

First step in the development of the application was creating an entry point. This is done by a main file that executes certain code when the HTML file is loaded, i.e. when the DOM is loaded. This can be seen in Listing 4.15. In here, we make sure that all the used

JavaScript libraries like jQuery, jQueryUI and so on are loaded via RequireJS[1] before we start the actual execution of the application. RequireJS helps us to ensure that all JavaScript gets loaded before we try to execute anything. When everything is loaded, we call `core.init()`.

4.2.4 Core

The core module is the beating heart of the application handling everything. From making sure everything gets loaded in the application to the actual communication between the modules. The structure of the core module can be seen in Figure 4.8. Every module in the application is expected to use the core module. It allows modules to dynamically load in new modules, extract data from the user, communicate with other modules and so on. We will explain more details about the core in the next sections.

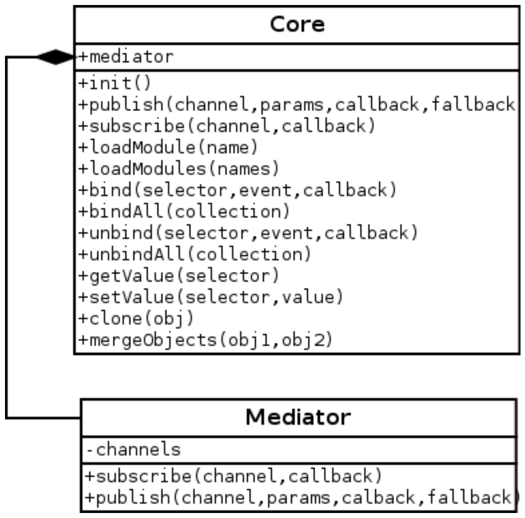

Core
+mediator
+init()
+publish(channel,params,callback,fallback
+subscribe(channel,callback)
+loadModule(name)
+loadModules(names)
+bind(selector,event,callback)
+bindAll(collection)
+unbind(selector,event,callback)
+unbindAll(collection)
+getValue(selector)
+setValue(selector,value)
+clone(obj)
+mergeObjects(obj1,obj2)
Mediator
-channels
+subscribe(channel,callback)
+publish(channel,params,calback,fallback

Figure 4.8: UML diagram showing the core and mediator, the beating heart of the application

4.2.4.1 The mediator: communication between modules

We placed the code into modules which gave rise to a question: how do we let modules communicate with each other? As we mentioned before, a web application is in fact a whole bunch of functionality that reacts to events like the user clicking on a button. We took this event-based approach into our module system. Each module is able to raise events and to react to events. What is actually implemented, is a sort of publish-subscribe architecture [23] but with a twist. It is based on publish-subscribe architecture ideas like *future type message* [77, 2] and *wait-by-necessity* [67].

Future type message and wait-by-necessity are less restrictive variants of the concept *fire-and-forget*. In fire-and-forget, invocations are made without expecting a result. If

[1]`http://requirejs.org/`

we would use that concept in our application, a module might make an invocation or raise an event without ever reacting to a possible result.

In future type message and wait-by-necessity, there is support for return values. These return values are actually handles which the developer can use to access the actual result of a method invocation some time in the future. If we would use this concept, a module would be able to react in the future. We twisted the architecture a little bit by allowing the developer to add a callback to their event raise.

Subscribing to events Upon starting the application, certain modules are loaded. By convention, every module can have a `init` method that gets called automatically as the module gets loaded. In this `init` method, modules can let the core know that they want to subscribe to an event. An example of such an `init` method is shown in Listing 4.16. By calling the subscribe method of the core, modules notify that they want the given callback to get called whenever the event occurs. Via this, we cover the subscribe-part of the publish-subscribe architecture.

```javascript
var init = function() {
  core.subscribe('editPresentation', showScreen);
  core.subscribe('presentationCreated', addTab);
  core.subscribe('presentationOpened', addTab);
  core.subscribe('allPresentationsClosed', destroy);
  core.subscribe('initPresentationTabs', initTabs);
  core.subscribe('allPresentations', gotAllPresentations);
}
```

Listing 4.16: An example of an init method of a module

Publishing events We will explain the publishing of events more in detail via an example in Listing 4.17. In this example, we publish that we want to receive some data about a presentation element. This is a common situation where we want to know what the result is as we need it to continue our computations. The first parameter is the *message* by which other modules can know if they can react to it. The second parameter is a collection of extra parameters that the possible receiving module can use. The third parameter is the callback which will be executed on the result of every possible module that returns a result upon invocation. It is thus not known upfront how many modules might react to the given message. This is part of the flexibility. A final parameter is available which is called the fallback. This is a closure that gets called whenever no module was able to react to the given event.

```javascript
core.publish('getPresentationElementData', element, function(element)
    {
  // do something
});
```

Listing 4.17: Basic example of how the publish in our publish-subscribe architecture works

4.2.4.2 The core: glue between HTML and JavaScript

Apart from handling all possible subscriptions on events and publishes, the core is also responsible for providing an easy-to-use interface for the developers to get and set data.

Binding to DOM events In an HTML webpage, multiple events can occur. A user might click on a button, move their mouse, enter text and so on. All these events are captured by the user's browser and are called DOM events [1]. The DOM Events Specification allows developers to register to event handlers, access the event flow and provide contextual information for each event. For example the developer is capable of knowing where exactly a user might have clicked and what events are triggered because of this.

To allow modules to bind handlers to these DOM events via a generic and flexible interface, we provided the core with four methods: `bind`, `unbind`, `bindAll` and `unbindAll`. The `bind` method allows a developers to register a handle to a DOM event and he can unregister the handle via the method `unbind`. With `bindAll` and `unbindAll`, he can register and unregister a collection of events. All DOM event types are supported by the core.

Getting and setting data The user is constantly interacting with the webpage that they see. One of the possible interactions is filling in a form. To extract the information filled into a form, we provided the `getValue` method. Via this method, the developer has an easy to use interface to extract data. Thanks to the counterpart method `setValue`, they can also set certain data in the HTML.

4.2.4.3 Preparing the application to start

One of the major functionalities that the core provides, is helping with the start of the application. As can be seen in Listing 4.15, the main file calls the instance of the core with the `init` method. In this method, the following operations are performed sequentially: loading all modules, calling all `init` methods of the modules, loading all plugins and showing the starting screen to the user. Because all these actions are bundled into one method, we are sure that everything is loaded in correctly before the user ever gets to see the starting screen. This way we can ensure that everything can work correctly before the user starts performing any action.

[1] `http://www.w3.org/TR/DOM-Level-3-Events/`

4.3 Details about the implementation

In Chapter 3, we introduced the best possible tool for authoring MindXpres presentations and themes alongside possible improvements for current presentation tools. We now present what we actually implemented of the best tool in a first version, after we had implemented the JavaScript architecture that we presented in Section 4.2.

4.3.1 Used technologies and libraries

The application is entirely built in HTML5, CSS3 and JavaScript. As a result, the application can completely run in a web browser without any problem. It is not dependent on plugins. To get certain complex functionality, we used several libraries. The largest library we used is jQuery as we concluded in Section 4.1 that it is currently the best JavaScript library out there to build a user interface.

Of course, jQuery does not offer all the features that we needed. For the more user interface-centered features, jQuery relies on jQuery UI. jQuery UI is a curated set of user interface interactions, effects, widgets and themes built for the jQuery library. Functionality like *resize* and *draggable* becomes available to developers with jQuery UI.

One of the main features in MindXpres is the ability to draw paths between elements in a presentation. To achieve this, the functionality of connecting elements was necessary. Thanks to jsPlumb[1], this was easily achievable. We only had to customise the look-and-feel a bit to fit in the MindXpres application.

In order to get context menus in the application, we make use of the jQuery contextmenu library[2]. Note that our application is not completely dependent on it as most of the functionality is also reachable via other menus.

4.3.2 Implementations made

An overview of what we actually implemented is:

- Starting screen
- Support for help screen
- Creating presentation
- Opening presentation
- Creating theme
- Opening theme
- Adding presentation elements
- Filter possible presentation elements to add
- Moving presentation elements

[1]`http://jsplumbtoolkit.com/jquery/demo.html`
[2]`https://github.com/medialize/jQuery-contextMenu`

- Resizing presentation elements
- Deleting presentation elements
- Contextmenu on presentation elements
- Tabs to edit multiple presentations at the same time
- Basic minimap when editing presentations
- Zooming in presentations
- Add path element
- Move path element
- Delete path element
- Contextmenu on path elements
- Show and hide presentation path
- Edit theme of the presentation currently editing
- Exporting presentation elements into a theme
- Add theme element
- Filter possible theme elements to add
- Edit theme element
- Delete theme element
- Filter theme elements
- Tabs to edit multiple themes at the same time
- Applying a theme to presentations
- Use theme in a new presentation

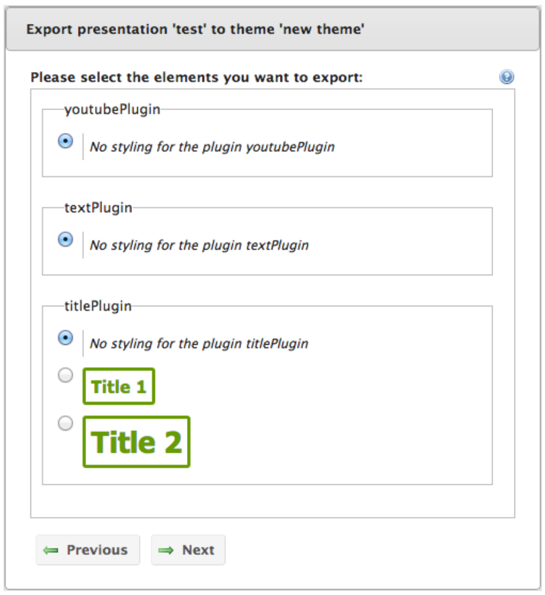
4.3.3 Some important implementations highlighted

We will now discuss certain important parts of the application that we developed. These are either interesting concepts that we implemented or they offer a better understanding of how the application generally works.

4.3.3.1 Exporting presentation elements into a theme

This feature is one of the four connecting parts between the presentation editor and the theme editor. The idea behind it is fairly simple: when users are editing a presentation, they are editing the stylings of elements. For example they might add a title element and change the font size. The user might want to save these changes into a theme. To avoid that they have to redesign a theme from scratch and reproduce such new stylings, we offer the necessary interfaces to export the existing stylings into a theme.

Firstly we added an easy-to-use button to the interface by which the user makes clear that they want to export the stylings into a theme. After they click on it, a dialogue appears in which the user can fill in the new theme name and other information. When they click next, a new dialogue appears in which they can select what elements they wants to export as can be seen in Figure 4.9. After that, they get an overview of the selected elements and they are able to either create the theme or return back to make changes.


Export presentation 'test' to theme 'new theme'

Please select the elements you want to export:

youtubePlugin

[✓ No styling for the plugin youtubePlugin]

textPlugin

[✓ No styling for the plugin textPlugin]

titlePlugin

[✓ No styling for the plugin titlePlugin]

Title 1

Title 2

Figure 4.9: The user gets an overview of all the elements in the presentation. In the current presentation, we only added two title elements with different font sizes which is shown to the user.

This is an example of the flexibility of the plugins in the MindXpres application. Each presentation consists of a bunch of plugin instances. Whenever the user changes the styling of a certain instance, this is saved in the instance itself obviously. In order to export the stylings, we only have to take all the presentation elements and sort them on their type. When they are sorted, we only have to show the correct selection boxes in a dialog and we can show the actual plugin instances simply be asking them to render.

The actual export is basically a copy of the plugin instances from the presentation into a new theme. Hence a theme will consist of a collection of plugin instances, just like a presentation. The difference is that instances in a theme are unique based on their type. A theme will have a maximum amount of one title instance, one text instance, one YouTube instance and so on.

4.3.3.2 Tabs to edit multiple presentations at the same time

To allow the user to edit multiple presentations at a time, we introduced tabs via which the user can change the current presentation that he is editing. By default, jQuery supports the addition of tabs in a webpage, yet we implemented them ourselves to have full control.

The functionality of tabs for presentation editing is embedded in a single module. This module reacts to all possible events that might change the tabs. For example when a new presentation is created, a tab must be added. To react to the situations when a new tab must be added, we only had to subscribe to the appropriate events as can be seen in Listing 4.18.

```
core.subscribe('presentationCreated', addTab);
core.subscribe('presentationOpened', addTab);
```

Listing 4.18: By adding two basic lines it was possible to have new tabs added at the same time as when presentations were opened and created

This is an example of how flexible the module system is. The code that creates or opens a presentation does not have to be altered to allow the correct appearance of tabs. It is perfectly possible to disable the module that manages the tabs and simply limit the user to having only one presentation at the same time opened. The tabs functionality is in fact injected into the already working presentation tool. A result of opening and creating multiple presentations can be seen in Figure 4.10.

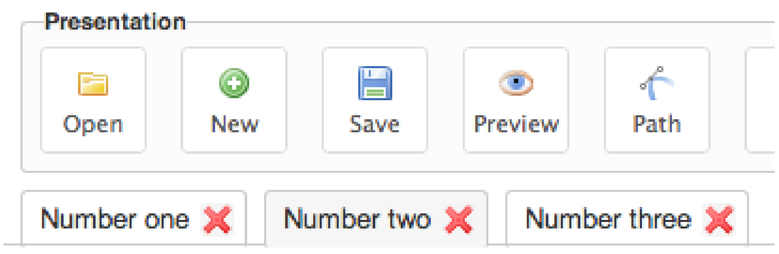
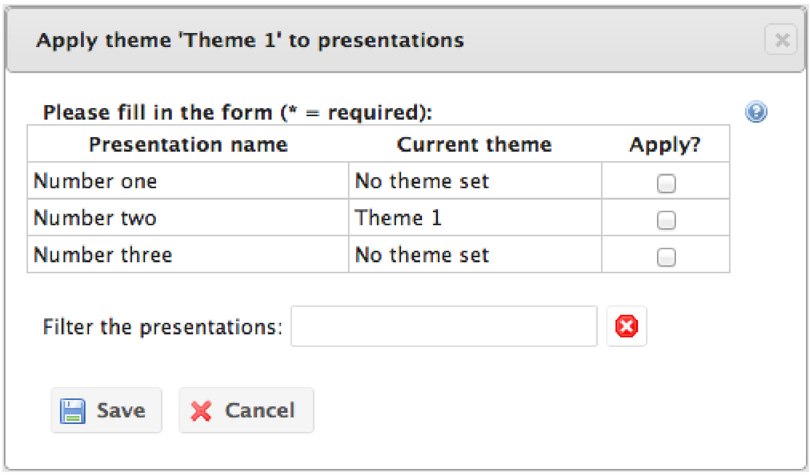


Figure 4.10: As the user opens or creates multiple presentations, tabs are added to allow him to change the presentation that he is currently editing

4.3.3.3 Applying a theme to presentations

One of the four links between the presentation and theme editors was the ability to apply a theme to presentations without having to open all these presentations. Because we use transclusion, this process is actually fairly easy. Theme specific stylings are never saved in a presentation. The presentation only keeps a reference to the theme they use. Each theme can also point to another theme which results in a whole chain of stylings.

As a presentation only keeps a reference, we only have to overwrite the reference in the presentation. It is only at viewtime that a presentation gets rendered and that theme definitions get loaded. To implement the update of the reference, it was only a matter of setting the member *theme* of a presentation object to its correct value.



Apply theme 'Theme 1' to presentations

Please fill in the form (* = required):

Presentation name	Current theme	Apply?
Number one	No theme set	☐
Number two	Theme 1	☐
Number three	No theme set	☐

Filter the presentations:

Figure 4.11: Via a dialog, the user can select on which presentations he wants to apply a certain theme.

4.4 Evaluation

The final result consists of over 8000 lines of code. We tried to implement as much of the desired functionality as possible to show the actual power of the proposed presentation tool. The result was a real stress test for the proposed JavaScript architecture and the used libraries.

In our JavaScript library comparison that we performed in Section 4.1, we came to the conclusion that jQuery would be the best library out there to help us build this application. jQuery offered out of the box lots of functionality and thanks to side projects like jQuery UI, we had no trouble at implementing a lot of features. For example the ability to resize elements was simply a matter of activating the *resize* functionality on the correct HTML elements. After implementing such a large project, we can conclude that it really was the best choice.

The proposed JavaScript architecture that we used, evolved over time. As we first started, we did not have the ability to provide callbacks when a module published something. This meant that if a module could respond to a certain publish (i.e. he subscribed to the corresponding message) that it as impossible to directly react. The return values were ignored as we wanted to have the modules as loose coupled as possible.

At first this worked well and in certain cases it was no problem that there were no return values. For example the addition of tabs like we explained in Section 4.3.3.2 was easy to achieve as the tabs module only changed HTML and did not have to respond to other modules. It only had to perform certain actions whenever a presentation was opened or created.

However, certain modules needed the possibility to return values. A good example for this are the so called *manager* modules. These are modules that manage objects for a certain type like for example all the presentation objects. We introduced these modules to combine all the functionality that is necessary to manage these objects. However sometimes other modules would ask the manager modules to give back a certain object like for example a specific presentation object. As there were no return values in our first design, the manager modules simply published the object and the original requesting module had to subscribe to that specific publish to retrieve the object.

```javascript
var newPresentationElement = function(element) {
    core.publish('getPresentationElementData', element, function(
       elementWithData) {
      // do something with elementWithData
    });
}
```

Listing 4.19: Example usage of the callback

Although this works fine, the problem is that it lacks maintainability. To solve this, we added the callback that can be added when one publishes something. This callback will get called for any possible response with a correct return value as the first parameter. An example to explain this can be found in Listing 4.19. Here we see a method that reacts whenever a new presentation element is added to a presentation. It performs a publish that will be captured by the presentationelement manager module. Via the callback, we get back a presentation element with all possible missing data filled in. This code is used in the overviews module which gives a small overview in the sidebar of all the different presentation elements in a presentation.

Because of improvements like this, we were able to create a solid JavaScript architecture. In this architecture, modules are extensively used and they can communicate with each other while still being loose coupled. It is therefore not necessary for a module to be active to have a working application. When a module responds can even be variable. The architecture can be extended to use asynchronous calls to a backend for example.

We conclude that the proposed JavaScript architecture works well. We were able to create flexible modules that are loose coupled and in future even can be improved to use asynchronous calls. The choice of jQuery was the right choice and thanks to some good JavaScript libraries we were able to produce a huge amount of code in a very short timespan.

5 Use case

After designing a better presentation tool in Chapter 3, we made an implementation that we discussed in Chapter 4. In this chapter we will go through a possible use case of the presentation tool. We designed a scenario that will highlight the major benefits of our tool by going through the different novel concepts that we introduced.

5.1 Starting the application

As the application is a Rich Internet Application, it must be opened in the user's browser. Upon launching the application, the user is presented with a basic starting screen which allows them to create and open presentations and themes as can be seen in Figure 5.1.

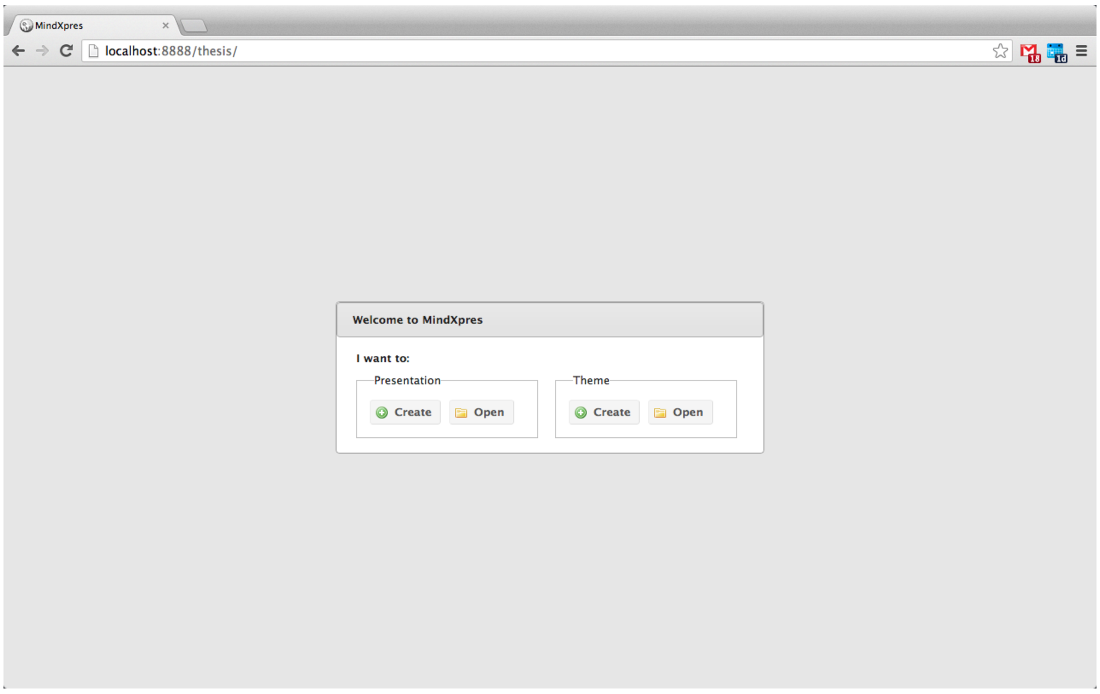
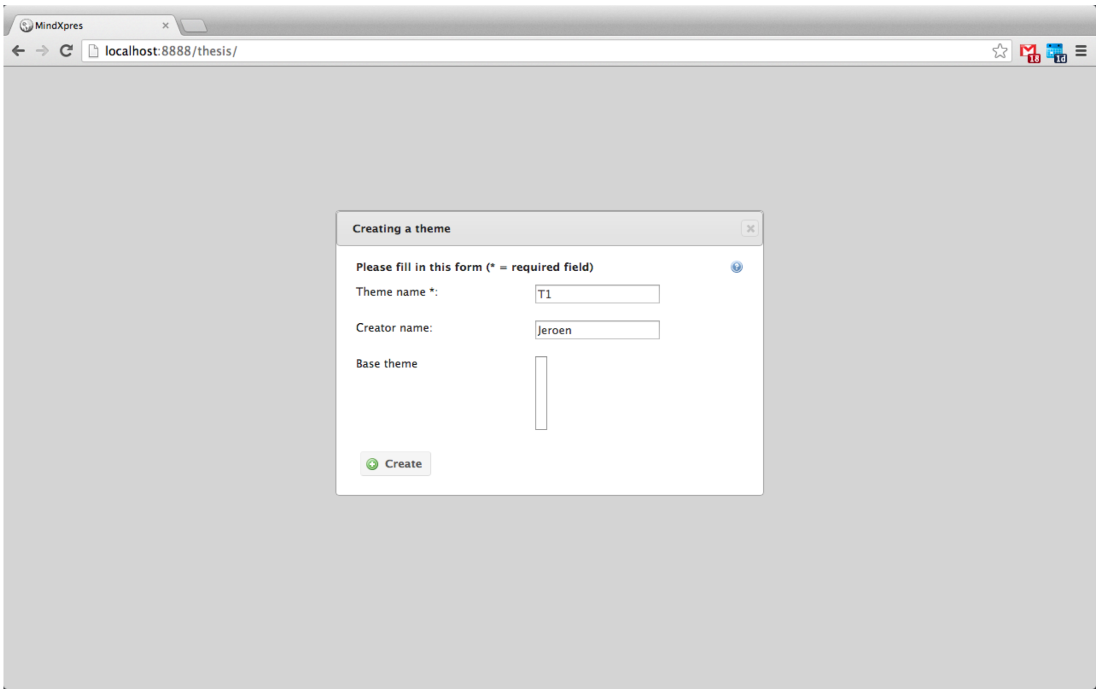


Figure 5.1: The starting screen that is presented to the user upon launching the application

5.2 Creating a theme

The tool focuses on the authoring of themes and presentations. We will start with creating a theme that will act as the basis for the styling of future presentations. The user can click on the `Create` button on the starting screen after which a dialogue is presented as shown in Figure 5.2.



MindXpres

localhost:8888/thesis/

Creating a theme

Please fill in this form (* = required field)

Theme name *:	T1
Creator name:	Jeroen
Base theme	

Create

Figure 5.2: The dialogue that allows the user to create a theme

5.3 Adding a theme element

A theme is a collection of theme elements. Each theme element holds the styling definitions for a specific type of element. In our current version, the user can have three different theme elements, one for each type of element that they can put in a presentation: `text`, `title` and `YouTube`. We added a theme element for the `text` type by clicking at the top of the application on the menu button `Add` in the group `Theme element`. The user is presented with a menu to choose which theme element he wants to add. In this scenario we added a text element. The result is a box in the interface that represents this theme element and it is illustrated in Figure 5.3.

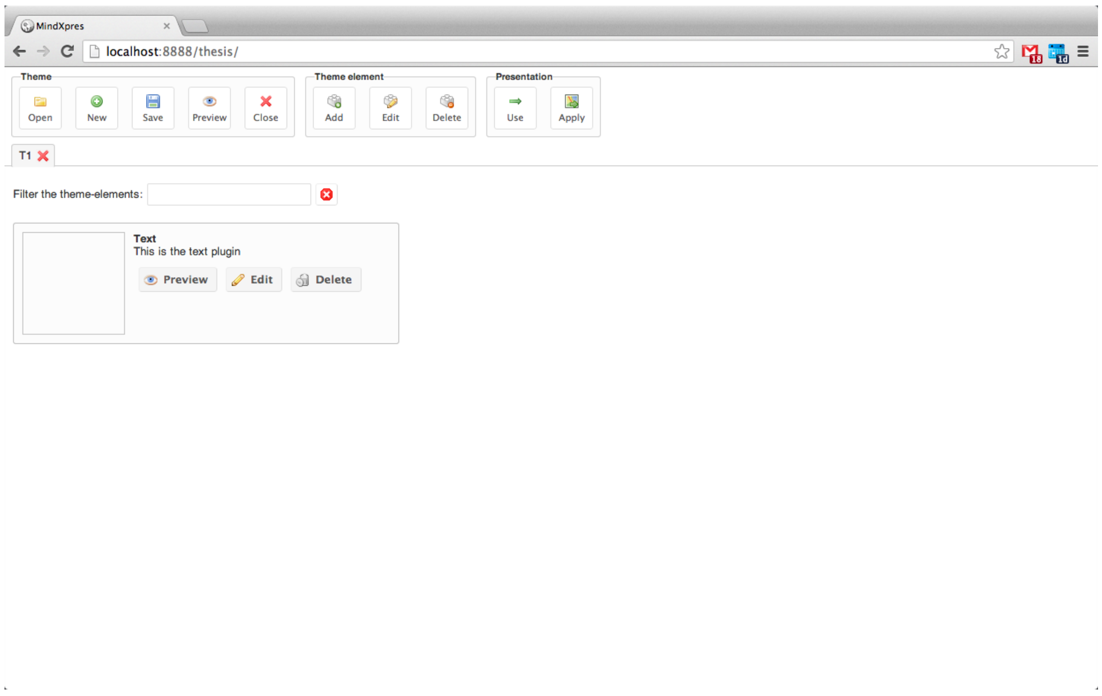


Figure 5.3: The result of adding a theme element to a theme

5.4 Editing a theme element

By clicking on the `Edit` button in a theme element box, the user is presented with a screen in which he can edit certain settings for that specific element, as can be seen in Figure 5.4. These settings are generated based on what is defined in the original plugin that renders the theme element. As the user is changing the settings, the small preview in the dialogue is updated instantly. We change the font size to the maximum size. By clicking on the `Save` button, the changes are saved in the theme.

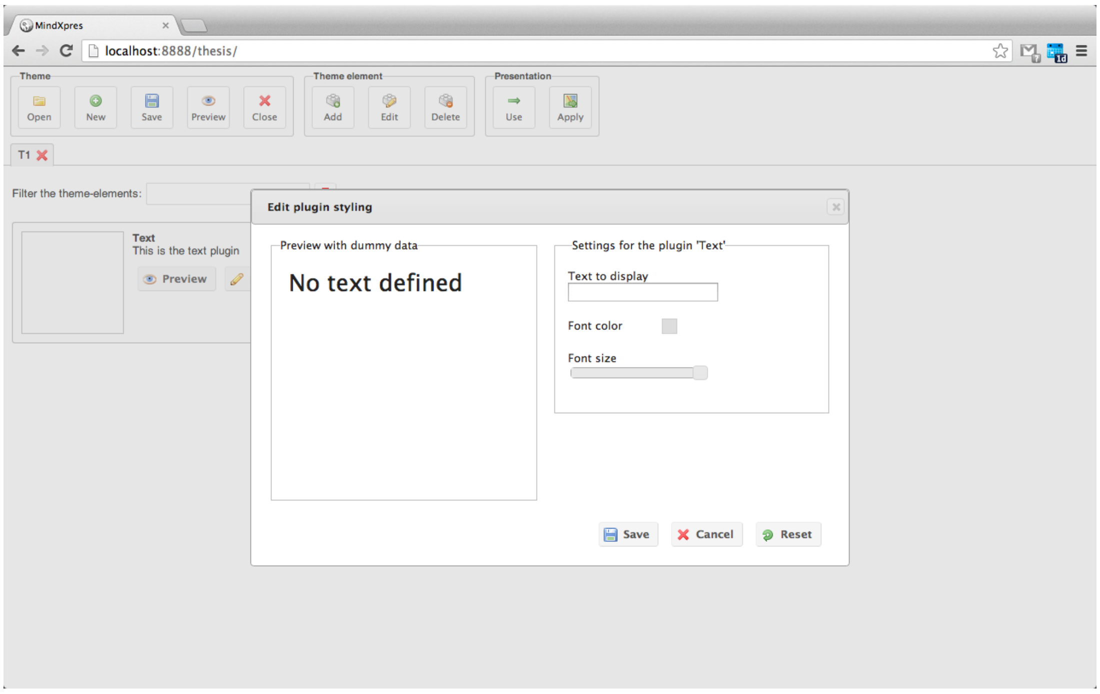
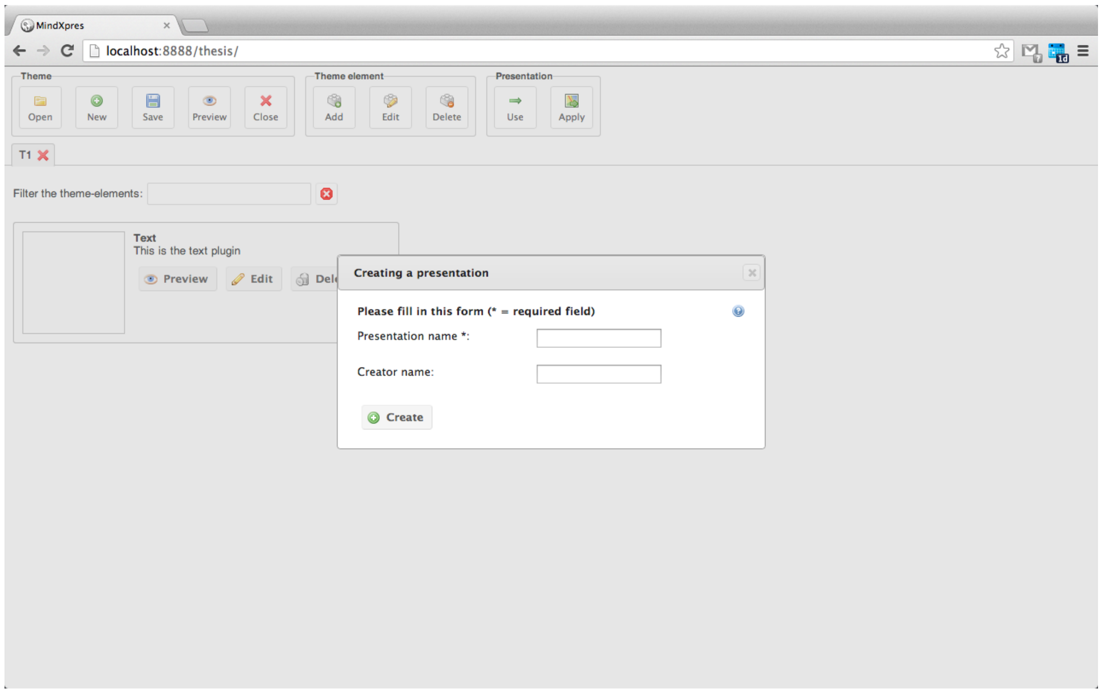


Figure 5.4: The dialogue that allows the editing of a theme element

5.5 Using a theme in a new presentation

One of the novel concepts that we introduced in our presentation and theme authoring tool is the possibility to directly create a new presentation from within the theme editor. This can be achieved by clicking on the `Use` button in the group `Presentation` which is situated in the top of the application. The user is presented with a dialogue, as illustrated in Figure 5.5. Note that this is almost an exact replica of the dialogue that was shown in Figure 5.2 with the exception of the lack of being able to select the theme. This selection of the theme is already done for the user. As soon as they click on the `Create` button, they are presented with the presentation editor with their newly created presentation opened.



Filter the theme-elements:

Text
This is the text plugin

Creating a presentation

Please fill in this form (* = required field)

Presentation name *:

Creator name:

Figure 5.5: The user is able to create a presentation from within the theme editor

5.6 Adding a presentation element

We will now add a presentation element to the presentation. This can be achieved in a very similar fashion like it was the case when adding a theme element. There is an `Add` button in the top of the application which gives a selection menu from which the user can choose the elements that he wants to add. After selecting an element, it is inserted in the presentation and rendered by the original plugin. We added a `text` element which is rendered by the `text` plugin. As we had a theme element for the `text` element, the theme stylings are applied to this new presentation element. The effect is that the text is in a larger font size than defined in the original plugin.

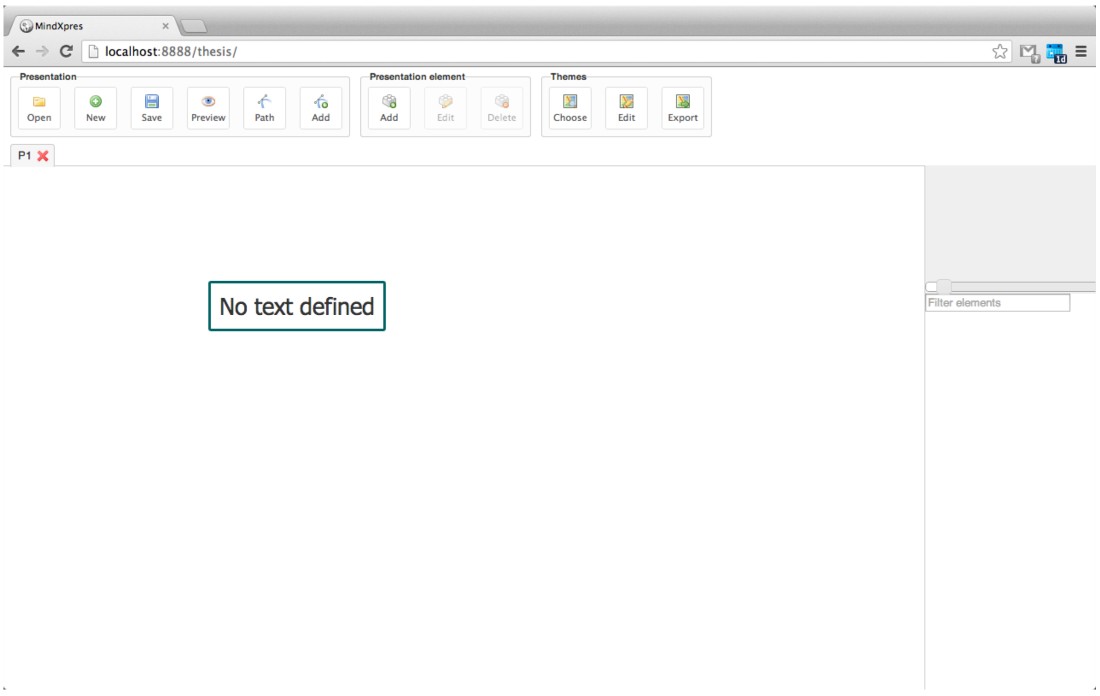


Figure 5.6: The result of adding a presentation element to a presentation. The element is rendered by the corresponding plugin.

5.7 Creating a new presentation

We close the presentation which results in an automatic save of the presentation itself. We are redirected back to the starting screen in which we click on the `Create` button to create a new presentation. We fill in the details and select our previously made theme T1 as the theme for our presentation.

5.8 Editing the current theme

After we entered the presentation editor by creating the presentation, we click on the `Edit` button in the `Themes` group in the top of the application. This is the second of the four novel concepts that we introduced. It will directly close the presentation editor and open the theme editor with theme T1 opened. The user is then able to edit theme T1 which will have an effect on all the presentations that use T1 as their theme. In this scenario we close theme T1.

5.9 Exporting theme elements

We will now create a new presentation P2 with several elements of the type `title`. When we double click on each of these elements, we change the settings so that each of them have a different font size. The result is illustrated in Figure 5.7.

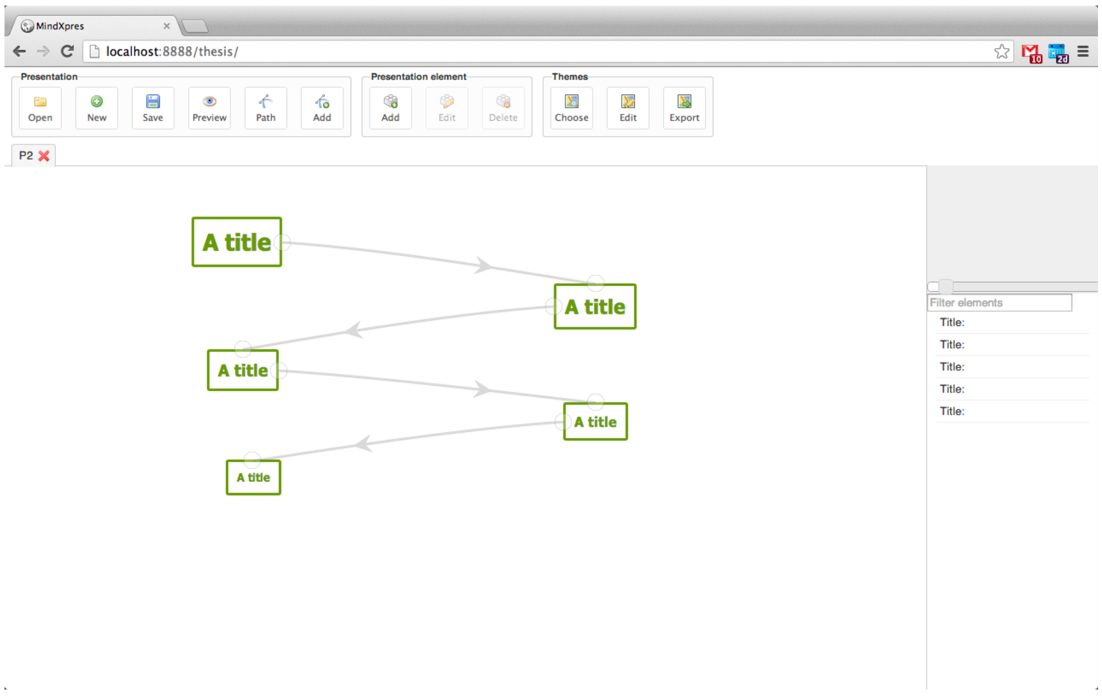
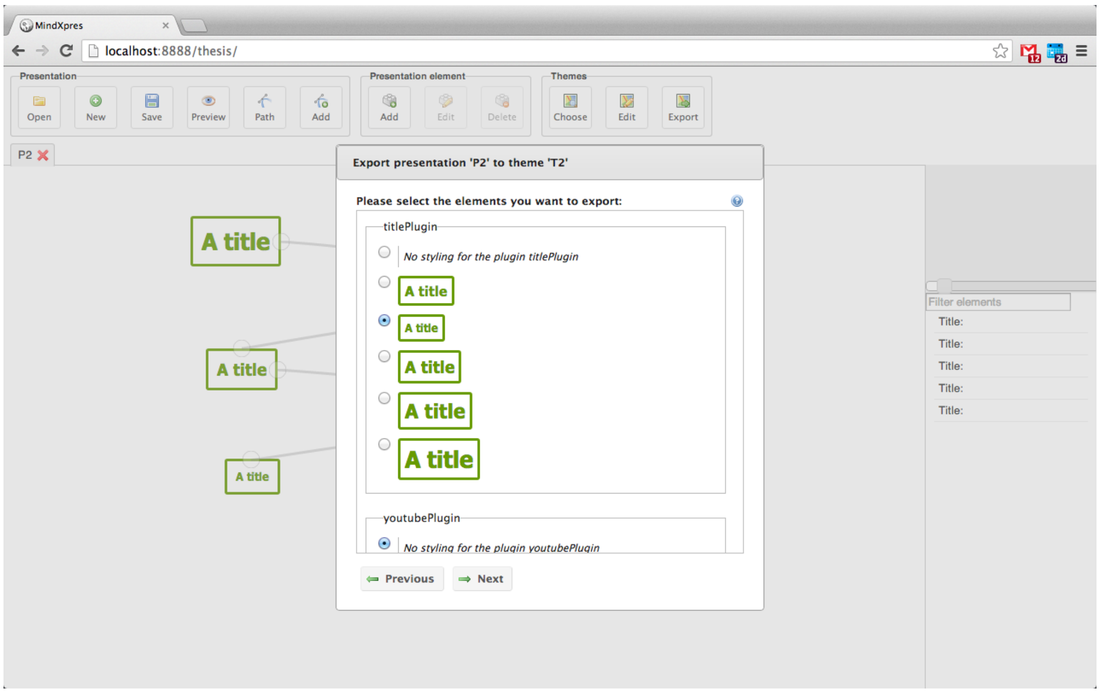


Figure 5.7: The presentation from which we will export stylings into a new theme

Now in the case that we want to save the stylings from one of these presentation elements, we are able to use the newly introduced export functionality. This is one of our four novel concepts which allows the extraction of presentation element stylings to be saved into a new theme. We simply click on the `Export` button at the top of the application after which we get a basic dialogue. In this dialogue, we can define the new theme name and other information like the author. When we click on next, we get the possibility to select a presentation element for each type of plugin as can be seen in Figure 5.8. After we selected the element that we want to export, we click on the next button after which we can review our selection and confirm to the application that the theme may be created.



Export presentation 'P2' to theme 'T2'

Please select the elements you want to export:

titlePlugin

- ○ *No styling for the plugin titlePlugin*
- ○ A title
- [✓ A title]
- ○ A title
- ○ A title
- ○ A title

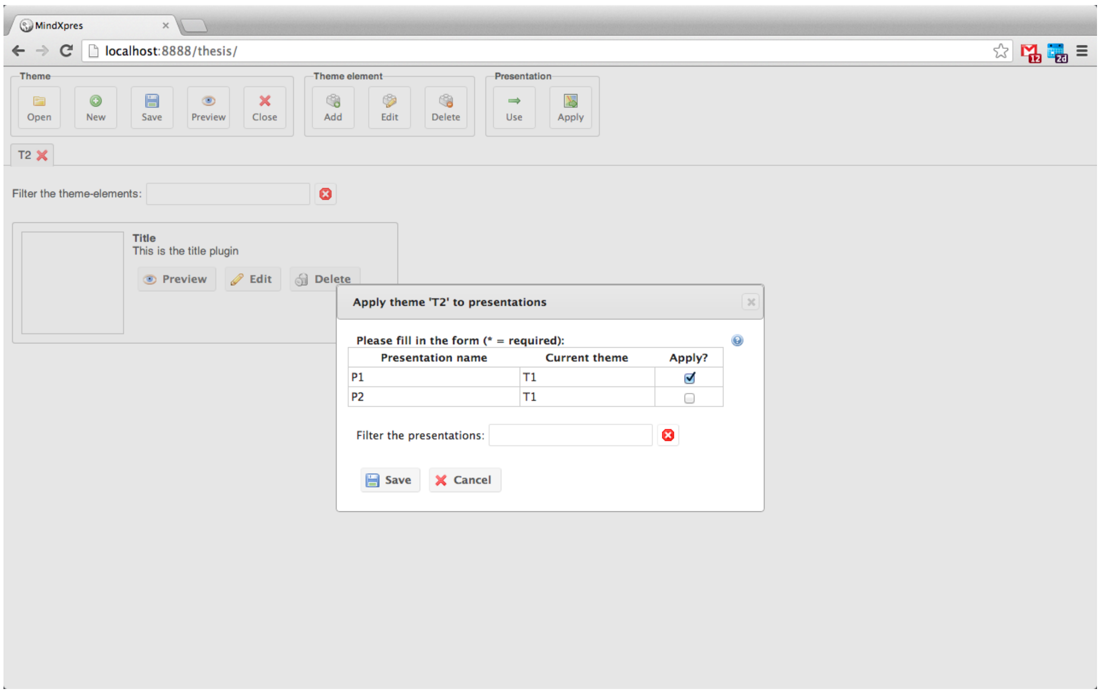
youtubePlugin

- [✓ *No styling for the plugin youtubePlugin*]

Figure 5.8: The user gets an overview of all the elements he can export

5.10 Applying a theme to a presentation

We now have a new theme T2. If we would want to change the theme of presentation P1 in the current popular presentation tools, we would have to open the presentation and go select the theme. In MindXpres, we introduced a novel method that allows the user to apply a theme to a presentation. We have theme T2 opened in the theme editor. When we click on the `Apply` button, we get the dialogue that is shown in Figure 5.9. In this dialogue the user has the possibility to select to which presentations the theme must be applied to. We select presentation P1 and after clicking on `Save`, the base theme for P1 is changed from theme T1 to T2.



Apply theme 'T2' to presentations

Please fill in the form (* = required):

Presentation name	Current theme	Apply?
P1	T1	☑
P2	T1	☐

Filter the presentations:

Save Cancel

Figure 5.9: Via a dialogue with an overview of all the presentations, the user can select to which presentations a theme must be applied to

6 Conclusion

In this final chapter, we summarise our findings from Chapter 2 about the current presentation tools and how they could be improved. We also recapitulate from Chapter 3 what is the best possible tool that could be designed for MindXpres. Finally, we take a look back at what we implemented in Chapter 4 and what conclusions we can draw from our experiences.

6.1 Evaluation

The main target of this thesis was to come up with a WYSIWYG authoring solution for the MindXpres presentation tool. During the course of this year, we have gone through different phases of the project. We started with a comparison of the current presentation tools. After that, we decided to go for a web-based solution which lead us to a comparison of current JavaScript libraries. With the results of these comparisons in mind, we started to design the best possible presentation and theme editor for the MindXpres presentation tool. To achieve this, we have gone through a good amount of literature by which we would be certain that particular design choices were the right ones. By creating mockups, we designed a presentation tool and came up with new and interesting concepts that are also applicable in current presentation tools. These new concepts could benefit the users of current presentation tools. We ended with building a first functional version of the presentation and theme editor.

6.1.1 Comparing current presentation tools

Although PowerPoint is the most used presentation tool with an estimated market share of about 95%[1], several alternatives exist. Apple Keynote and OpenOffice Impress are the most known alternatives for Mac- and Linux-users. Therefore, we took these two into our comparison with PowerPoint. We also looked at Prezi, a very promising and novel presentation tool that stepped away from the sequential slide paradigm.

While comparing these presentation tools, we divided the feature comparison into two categories: presentation editing and theme editing. Although most tools support the functionality to edit presentations and themes, we noted that this separation is not always that clear. For instance, we could not find a way to create a theme without having to create a presentation first. Most tools also have the exact same interface for editing themes as for editing presentations. Although this can be a positive thing as it gives a consistent interface, it can be confusing. Another problem is that multiple names are used to define what is in fact a theme: template and master slides are the most common ones.

Another problem that we identified within almost all presentation tools are the inconsistent interfaces for the functionalities. For example, PowerPoint offers multiple interface concepts to the user to activate certain functionality: via the ribbon, via a general menu bar at the top of the screen and so on. Although this gives the user the freedom to use whatever interface he likes, the functionalities available are not the same in all these interface concepts. We conclude that they lack consistency. A tool like Prezi does it much better as they provide only one interface to add for example a presentation element.

With the identified problems, we not only tried to solve these existing problems but also

[1]`http://www.businessweek.com/articles/2012-08-30/death-to-powerpoint`

tried to come up with novel solutions that are completely new in presentation tools. In total we introduced four new solutions:

- Using a theme directly for creating a presentation
- Editing the theme of a presentation
- Applying a theme to multiple presentations
- Extracting styling information from presentations into a theme

These four novel concepts are in fact links between a separate theme and presentation editor. The problem with separating a presentation tool into two different editors is that the user experiences a lack in freedom to switch between the editors. The four novel concepts provide two links in each direction which gives the user the complete freedom to switch back and forth.

6.1.2 Comparing JavaScript libraries

As the main goal of this thesis was to implement a theme and presentation editor for the MindXpres presentation tool, we had to look at what technologies we were going to use. As the web continues to evolve with new standards like HTML5 and CSS3, it was certainly an option to implement the presentation tool as a rich internet application. Other motivations to implement a rich internet application entirely in HTML5, CSS3 and JavaScript were the platform independence and not having to install additional software like browser plugins.

However, to create such a vast application like a presentation tool, we wanted to use a JavaScript library that would support lots of functionality by default that we would use often. Things like draggable elements, resizable elements and helper functions for strings and objects were one of the many features that we wanted to have available by default. Implementing such features ourselves would be possible but it would lengthen the implementation time drastically.

Thus it was interesting enough to check out the current state-of-art of JavaScript libraries. In total we found five commonly used JavaScript libraries with lots of features available. Deciding which of these libraries would be the best for our presentation tool implementation was tricky. We started out with a feature comparison which showed us that all five of them are very competitive.

After the feature comparison, we started to use the libraries themselves. How easy was it to get the libraries running and create a basic *Hello world!* application? We concluded that all of them were almost equally competitive.

The real differences started to show at our third level of comparison: a basic implementation of a webpage with certain features. Initially we wanted to start with a basic webpage that we would continue to extend with functionalities to stress test the libraries. Starting with a webpage were we had an element that was draggable and an element that was resizable, all libraries were still competitive. However, when we decided to combine these basic functionalities, only one library was able to achieve the desired result: jQuery.

What we learned from this comparison is that although libraries have very promising features, these features work very well when they are the only features used on a webpage. As soon as you start combining functionalities, the libraries start to fail. This is either because the functionalities override certain things or because they are very destructive on the original HTML.

jQuery was the best out of the five libraries that we compared. Not only did it present us with the expected result, it also did it with very clean and small code. Hence, choosing jQuery as the main library for our application was a trivial choice.

6.1.3 Designing a presentation tool

One of the key goals of this thesis was designing a theme and presentation editor for the MindXpres presentation tool. We created mockups that we used as a guideline for the actual implementation. They are also used to better explain certain concepts.

While designing the presentation tool, we based ourselves on lots of literature and previous studies to try and make the best design choices possible. The results from previous studies helped us to make certain choices. An example can be found in the interface to add elements. There were two different solutions to how a user can add an element. The first one is via clicking on the type of element that he wants and then clicking on the spot where he wants to place the element. The other one is via selecting a new element out of a list of types and dragging it onto the spot where he wants to place it. Thanks to previous studies, we could conclude that it depends on the type of input device whether a user prefers a point-and-click interaction or a drag-and-drop [47]. Hence we could make certain design choices or we could suggest possible solutions that could be taken into account at implementation time.

6.1.4 Implementing a presentation tool

The actual implementation of the MindXpres presentation and theme editors proved to be very challenging. Firstly we started with designing a good JavaScript architecture in which we would organise our code and functionalities. Secondly, we tried to abstract as much of the functionalities into a core and plugin core. After that, we started with the actual implementation of all the screens and the links between them.

Because we worked with modules that are able to communicate with each other, we were soon writing out lots of code that was nicely organised and worked cleanly together. The result was that after a couple of months of implementation, we ended up with an application that has more than 8000 lines of codes and offers a huge set of functionality.

6.2 Future work

6.2.1 Completing the presentation tool

As a master thesis, it was not expected that we could finish the whole implementation phase of the presentation tool. This simply was not realistic as it requires a lot of work. So, a first goal for future work is to continue the implementation phase in order to complete the actual implementation of the presentation tool.

6.2.2 Evaluating the tool

Due to time constraints, we were never able to perform real user tests where we see if our given solutions to certain design problems were effective. Because we based our design decisions mostly on previous studies, we are fairly confident that everything will turn out okay. Still, most of these previous studies were never used within the context of a presentation tool like MindXpres. It would be certainly wise to perform extensive user testing to find out whether certain design solutions were the best choice.

6.2.3 Extending the use of technologies internally

At the moment, the proposed and implemented MindXpres presentation tool uses few technologies: HTML5, CSS3 and JavaScript. Users have to open an HTML page to start the application. A possible step in future work could be to take a look at what technologies are available for a version that relies on things like server-sided technologies. For example, could the application be migrated into a typical website with a backend on the server that saves every step the user goes through?

6.2.4 Collaborative editing of presentations and themes

Some presentations are built in group and some themes are used by multiple people. In such case, it would come in handy to see how collaborative editing could be achieved. Can the application be turned into a presentation editing tool like the one available on Google Docs[1]? How can the editing be improved instead of just sending out the changes in real time? What are the considerations for when the network temporally fails? These are all questions that make it a very challenging assignment to move towards a MindXpres editor with collaborative editing.

[1] `https://drive.google.com/`

6.2.5 Investigating new input and output methods

Currently the presentation tool was designed to be mainly used with a mouse and keyboard while the user sits behind a computer. Because it fully runs on standardized web technologies, it should be possible to use it on a tablet for example. In such case, it would be interesting to see how the user actually uses the tool in comparison to when he sits behind a desktop or laptop. Also, other output methods like a projector could be considered. Different input methods like pen-and-paper, touchscreens and so on offer a possible better experience for the user. With possibly small adaptations to the tool, it can be beneficial for the user if he has the complete freedom to what device he is using with the tool.

6.3 Contributions

In Chapter 2 we performed a comparison on the currently popular presentation tools. We wanted to identify how they implemented the authoring of sequential slides in a presentation and what the problems are with this approach. We can conclude that the sequential slide paradigm limits the users in their creativity and flexibility. We listed these problems in Section 2.1. Some of these problems are already partially solved in a tool like Prezi, yet we were able to identify several other problems with this tool which we also presented in Section 2.1.

We also looked at how the concept of presentations and themes is linked in current presentation tools. One of the research questions that we had was how the concepts of themes and presentations are linked in current presentation tools and whether there are certain limitations to the current approaches. All of them use a non flexible approach as in most cases the theme definitions are copied into presentations. We propose a new approach via the concept of transclusion. In this concept, presentations and themes are completely separated and they are linked together in such a way that they can exist separately. At viewtime, the presentations and themes are merged together. This is inspired on how the Web currently works with HTML pages and media that is only fetched at viewtime.

Not only did we separate the concepts of presentations and themes in an efficient and flexible manner by using transclusion, we also presented several solutions on how to link the authoring of these concepts in a tool. We introduced four novel concepts that allow developers to have presentations and themes separated. Meanwhile the user still has maximum flexibility in authoring his presentations and themes.

We designed an authoring tool that was partially inspired by the current tools while we improved multiple existing concepts. We proposed a tool that works with a zoomable user interface, has a consistent menu structure and has a clear separation between the editing of presentations and themes. The editing of presentations and themes is nicely linked together by our four novel concepts.

As the Web continues to evolve, we looked at how we could use the Web as a new application platform. In order to have a dynamic webpage, developers are dependent on JavaScript technology. We performed a thorough comparison of the five major JavaScript libraries currently used on the World Wide Web. We identified several problems in these JavaScript libraries and came to the conclusion that jQuery is currently the best library to use. With this gained knowledge, we designed a flexible and solid architecture for our application that is inspired by the publish-subscribe design pattern and the AJAX concept.

We also did a basic comparison in Section 2.3 on the current user interface languages that looked promising. Sadly we came to the conclusion that none of them are directly usable in a Rich Internet Application.

As a final phase in this thesis, we implemented a first version of the authoring tool that we designed. We built it in JavaScript based on our JavaScript library comparison and the architecture that we designed.

6.4 Final words

In MindXpres, three major phases were identified: before, during and after giving a presentation. This thesis focused on the first of those three, the phase before giving a presentation. Based on the conclusions that we could draw on the current technologies, there is clearly room for improvement on both the design and implementation part. Yet it looks very promising for the future of MindXpres as we now have a solid basis to work with.

Bibliography

[1] Marc Abrams, C Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster. UIML: An Appliance-Independent XML User Interface Language. *Computer Networks*, 31(11):1695–1708, May 1999.

[2] Akkihebbal L. Ananda, B. H. Tay, and E. K. Koh. A Survey of Asynchronous Remote Procedure Calls. *SIGOPS Operating Systems Review*, 26(2):92–109, April 1992.

[3] Robert A. Bartsch and Kristi M. Cobern. Effectiveness of PowerPoint Presentations in Lectures. *Compututer Educaction*, 41(1):77–86, June 2003.

[4] Patrick Baudisch, Xing Xie, Chong Wang, and Wei-Ying Ma. Collapse-To-Zoom: Viewing Web Pages On Small Screen Devices By Interactively Removing Irrelevant Content. In *Proceedings of UIST '04, 17th annual ACM symposium on User Interface Software and Technology*, pages 91–94, Santa Fe, NM, USA, October 2004.

[5] David V. Beard and John Q. Walker II. Navigational Techniques To Improve The Display of Large Two-Dimensional Spaces. *Behaviour & Information Technology*, 9(6):451–466, 1990.

[6] Ben Bederson and Jon Meyer. Implementing A Zooming User Interface: Experience Building Pad++. *Software Practice & Experience*, 28(10):1101–1135, August 1998.

[7] Benjamin B. Bederson. Fisheye Menus. In *Proceedings of UIST '00, 13th annual ACM Symposium on User Interface Software and Technology*, pages 217–225, San Diego, California, USA, November 2000.

[8] Benjamin B. Bederson. The Promise of Zoomable User Interfaces. *Behaviour & Information Technology*, 30(6):853–866, June 2011.

[9] Benjamin B. Bederson and Angela Boltman. Does Animation Help Users Build Mental Maps of Spatial Information? In *Proceedings of INFOVIS '99, IEEE Symposium on Information Visualization*, pages 28–35, San Francisco, California, USA, October 1999.

[10] Benjamin B. Bederson, James D. Hollan, Ken Perlin, Jonathan Meyer, David Bacon, and George Furnas. Pad++: A Zoomable Graphical Sketchpad For Exploring Alternate Interface Physics. *Journal of Visual Languages & Computing*, 7(1):3–32, March 1996.

[11] Staffan Björk and Johan Redström. Redefining the Focus and Context of Focus+Context Visualizations. In *Proceedings of INFOVIS '00, IEEE Symposium on Information Vizualization 2000*, pages 85–89, Salt Lake City, Utah, USA, October 2000.

[12] Stefano Burigat, Luca Chittaro, and Edoardo Parlato. Map, Diagram, and Web Page Navigation on Mobile Devices: The Effectiveness of Zoomable User Interfaces With Overviews. In *Proceedings of MobileHCI '08, 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 147–156, Amsterdam, The Netherlands, September 2008.

[13] Vannevar Bush. As We May Think. *Atlantic Monthly*, pages 101–108, July 1945.

[14] Richard Cardone, Danny Soroker, and Alpana Tiwari. Using XForms To Simplify Web Programming. In *Proceedings of WWW '05, 14th International Conference on World Wide Web*, pages 215–224, Chiba, Japan, May 2005.

[15] Bay-Wei Chang and David Ungar. Animation: From Cartoons To The User Interface. In *Proceedings of UIST '93, 6th annual ACM symposium on User Interface Software and Technology*, pages 45–55, Atlanta, Georgia, USA, October 1993.

[16] Lorenzo Clementi. Focus and Context. Technical report, 2007.

[17] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A Review of Overview+detail, Zooming and Focus+context Interfaces. *ACM Computing Surveys*, 41(1):2:1–2:31, January 2009.

[18] Andy Cockburn and Joshua Savage. Comparing Speed-Dependent Automatic Zooming with Traditional Scroll, Pan, and Zoom Methods. In *Proceedings of HCI '03, Conference on People and Computers XVII: British Computer Society Conference on Human Computer Interaction*, pages 87–102, Bath, United Kingdom, September 2003.

[19] Douglas Crockford. *JavaScript: The Good Parts*. O'Reilly Media, Inc., 2008.

[20] Martin Dostál. User Acceptance Of The Microsoft Ribbon User Interface. In *Proceedings of DNCOCO '07, 9th WSEAS International Conference on Data Networks, Communications, Computers*, pages 143–149, Faro, Portugal, November 2010.

[21] Sarah A. Douglas and Arthur E. Kirkpatrick. Model And Representation: The Effect Of Visual Feedback On Human Performance In A Color Picker Interface. *ACM Transactions on Graphics*, 18(2):96–127, April 1999.

[22] Susan Dumais, Edward Cutrell, and Hao Chen. Optimizing Search By Showing Results In Context. In *Proceedings of CHI '01, SIGCHI Conference on Human Factors in Computing Systems*, pages 277–284, Seattle, Washington, USA, March 2001.

[23] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computation Surveys*, 35(2):114–131, June 2003.

[24] David K. Farkas. Toward a Better Understanding of PowerPoint Deck Design. *Information Design Journal*, 14(2):162–171, August 2006.

[25] Kenneth Feldt. *Programming Firefox: Building Rich Internet Applications with Xul*. O'Reilly Media, Inc., 2007.

[26] George W. Furnas. Generalized Fisheye Views. In *Proceedings of CHI '86, SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, Boston, Massachusetts, USA, April 1986.

[27] George W. Furnas and Benjamin B. Bederson. Space-Scale Diagrams: Understanding Multiscale Interfaces. In *Proceedings of the CHI '95, SIGCHI Conference on Human Factors in Computing Systems*, pages 234–241, Denver, Colorado, USA, May 1995.

[28] Robert Gaskins. Sample Product Proposal: Presentation Graphics for Overhead Projection, 1984.

[29] Robert Gaskins. PowerPoint At 20: Back To Basics. *Communications of the ACM*, 50(12):15–17, December 2007.

[30] Christian Geiger, Holger Reckter, Roman Dumitrescu, Sascha Kahl, and Jan Berssenbrügge. A Zoomable User Interface for Presenting Hierarchical Diagrams on Large Screens. In *Proceedings of HCI '09, 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, pages 791–800, San Diego, CA, July 2009.

[31] J. Gibb. JavaScript Smackdown: A Comparative Analysis of Web 2.0 Libraries. Technical report, 2006.

[32] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. Comparative Evaluation of Javascript Frameworks. In *Proceedings of WWW '12 Companion, 21st International Conference Companion on World Wide Web*, pages 513–514, Lyon, France, April 2012.

[33] Cleotilde Gonzalez. Does Animation in User Interfaces Improve Decision Making? In *Proceedings of CHI '96, SIGCHI Conference on Human Factors in Computing Systems*, pages 27–34, Vancouver, British Columbia, Canada, April 1996.

[34] Lance Good and Benjamin B. Bederson. CounterPoint: Creating Jazzy Interactive Presentations. *Human-Computer Interaction Laboratory, Institute for Advanced Computer Studies*, 3, 2001.

[35] Lance Good and Benjamin B. Bederson. Zoomable User Interfaces As A Medium for Slide Show Presentations. *Information Visualization*, 1(1):35–49, March 2002.

[36] Heiko Haller and Andreas Abecker. iMapping: A Zooming User Interface Approach for Personal and Semantic Knowledge Management. In *Proceedings of HT '10, 21st ACM conference on Hypertext and Hypermedia*, pages 119–128, Toronto, Ontario, Canada, June 2010.

[37] Kasper Hornbaek, Benjamin B. Bederson, and Catherine Plaisant. Navigation Patterns and Usability of Zoomable User Interfaces With and Without An Overview. *ACM Transactions on Computer-Human Interaction*, 9(4):362–389, December 2002.

[38] Kasper Hornbaek and Erik Frokjaer. Reading of Electronic Documents: The Usability of Linear, Fisheye, and Overview+detail Interfaces. In *Proceedings of CHI '01, SIGCHI Conference on Human Factors in Computing Systems*, pages 293–300, Seattle, Washington, USA, March 2001.

[39] Scott E. Hudson and John T. Stasko. Animation Support In A User Interface Toolkit: Flexible, Robust, and Reusable Abstractions. In *Proceedings of UIST '93, 6th annual ACM symposium on User Interface Software and Technology*, pages 57–67, Atlanta, Georgia, USA, October 1993.

[40] Takeo Igarashi and Ken Hinckley. Speed-Dependent Automatic Zooming for Browsing Large Documents. In *Proceedings of UIST '00, 13th Annual ACM Symposium on User Interface Software and Technology*, pages 139–148, San Diego, California, USA, November 2000.

[41] Kori M. Inkpen. Drag-and-Drop Versus Point-and-Click Mouse Interaction Styles For Children. *ACM Transactions on Computer-Human Interaction*, 8(1):1–33, March 2001.

[42] Jens E. Kjeldsen. The Rhetoric of PowerPoint. *International Journal of Media, Technology and Lifelong Learning*, 2(1):1–17, January 2006.

[43] Jonas Kluge, Frank Kargl, and Michael Weber. The Effects of the AJAX Technology on Web Application Usability. *3rd International Conference on Web Information Systems and Technologies WebIST 2007*, pages 289–294, March 2007.

[44] Avraham Leff and James T. Rayfield. Web-Application Development Using the Model/View/Controller Design Pattern. In *Proceedings of EDOC '01, 5th IEEE In-*

ternational Conference on Enterprise Distributed Object Computing, pages 118–127, Seattle, Washington, USA, September 2001.

[45] Leonhard Lichtschlag, Thorsten Karrer, and Jan Borchers. Fly: A Tool To Author Planar Presentations. In *Proceedings of CHI '09, SIGCHI Conference on Human Factors in Computing Systems*, pages 547–556, Boston, Massachusetts, USA, April 2009.

[46] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Víctor López-Jaquero. USIXML: A Language Supporting Multi-Path Development of User Interfaces. In *Proceedings of EHCI-DSVIS'04, International Conference on Engineering Human Computer Interaction and Interactive Systems*, pages 200–220, Hamburg, Germany, July 2004.

[47] I. Scott MacKenzie, Abigail Sellen, and William A. S. Buxton. A Comparison of Input Devices in Element Pointing and Dragging Tasks. In *Proceedings of SIGCHI '91, Conference on Human Factors in Computing Systems*, pages 161–166, New Orleans, Louisiana, USA, April 1991.

[48] Lori A. Macvittie. *XAML In A Nutshell: A Desktop Quick Reference*. O'Reilly Media, 2006.

[49] Jan Kasper Martinsen, Håkan Grahn, and Anders Isberg. A Comparative Evaluation of JavaScript Execution Behavior. In *Proceedings of ICWE '11, 11th International Conference on Web Engineering*, pages 399–402, Paphos, Cyprus, June 2011.

[50] Brad Mehlenbacher, Thomas M. Duffy, and James Palmer. Finding Information on a Menu: Linking Menu Organization to the User's Goals. *Human-Computer Interaction*, 4(3):231–251, September 1989.

[51] Tommi Mikkonen and Antero Taivalsaari. Using JavaScript As A Real Programming Language. Technical report, 2007.

[52] Arnab Nandi and H. V. Jagadish. Assisted Querying Using Instant-Response Interfaces. In *Proceedings of SIGMOD '07, ACM SIGMOD International Conference on Management of Data*, pages 1156–1158, Beijing, China, June 2007.

[53] Dmitry Nekrasovski, Adam Bodnar, Joanna McGrenere, François Guimbretière, and Tamara Munzner. An Evaluation of Pan & Zoom and Rubber Sheet Navigation With and Without An Overview. In *Proceedings of SIGCHI '06, Conference on Human Factors in Computing Systems*, pages 11–20, Montr&#233;al, Qu&#233;bec, Canada, April 2006.

[54] Theodor H. Nelson. Complex Information Processing: A File Structure For The Complex, The Changing And The Indeterminate. In *Proceedings of ACM '65, 20th National Conference*, pages 84–100, Cleveland, Ohio, USA, August 1965.

[55] Theodor H. Nelson. The Heart of Connection: Hypermedia Unified by Transclusion. *Communications of the ACM*, 38(8):31–33, August 1995.

[56] Den Odell. *Pro Javascript RIA Techniques: Best Practices, Performance and Presentation*. Apress, 2009.

[57] Ian Parker. Absolute PowerPoint. *The New Yorker*, 28:76–87, May 2001.

[58] Linda Dailey Paulson. Building Rich Web Applications With Ajax. *Computer*, 38(10):14–17, October 2005.

[59] Ken Perlin and David Fox. Pad: An Alternative Approach To The Computer Interface. In *Proceedings of SIGGRAPH '93, 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 57–64, Anaheim, California, USA, August 1993.

[60] Stuart Pook, Eric Lecolinet, Guy Vaysseix, and Emmanuel Barillot. Context And Interaction In Zoomable User Interfaces. In *Proceedings of AVI '00, Working Conference on Advanced Visual Interfaces*, pages 227–231, Palermo, Italy, May 2000.

[61] Michael J. Rees. Evolving The Browser Towards A Standard User Interface Architecture. In *Proceedings of AUIC '02, Third Australasian Conference on User Interfaces - Volume 7*, pages 1–7, Melbourne, Victoria, Australia, January 2002.

[62] Reinout Roels. MindXpres: An Extensible Content-driven Cross-Media Presentation Tool. Master's thesis, 2012.

[63] Ruth Rosenholtz. A Simple Saliency Model Predicts A Number of Motion Popout Phenomena. *Vision research*, 39(19):3157–3163, September 1999.

[64] Beat Signer and Moira C. Norrie. As We May Link: A General Metamodel For Hypermedia Systems. In *Proceedings of ER '07, 26th International Conference on Conceptual Modeling*, pages 359–374, Auckland, New Zealand, November 2007.

[65] Tad Simons. Does PowerPoint Make You Stupid. *Presentations, 18 (3)*, 24:6–11, March 2003.

[66] Bram Smeets, Uri Boness, and Roald Bankras. *Beginning Google Web Toolkit: From Novice to Professional*. Apress, 2008.

[67] Nice Sophia and Denis Caromel. Towards a Method of Object-Oriented Concurrent Programming. *Communications of the ACM*, 36(9):90–102, September 1993.

[68] Dudley Storey. *CSS3 Transforms and Transitions*. Springer, 2012.

[69] Kenneth L. Summers, Timothy E. Goldsmith, Steve Kuhica, and Thomas P. Caudell. An Experimental Evaluation of Continuous Semantic Zooming in Program Visualization. In *Proceedings of INFOVIS '03, Ninth annual IEEE conference on Information visualization*, pages 155–162, Seattle, Washington, July 2003.

[70] Anne M. Treisman and Garry Gelade. A Feature-Integration Theory of Attention. *Cognitive Psychology*, 12(1):97–136, January 1980.

[71] Edward R. Tufte. *The Cognitive Style of Powerpoint*. Graphics Press, 2003.

[72] Steven J. Vaughan-Nichols. Will HTML 5 Restandardize The Web? *Computer*, 43(4):13–15, April 2010.

[73] VisionMobile. Developer Economics 2013: Developer Tools: The Foundations of the App Economy, 2013.

[74] Helen J. Wang, Alexander Moshchuk, and Alan Bush. Convergence of Desktop and Web Applications on a Multi-Service OS. In *Proceedings of HotSec '09, 4th USENIX Conference on Hot Topics in Security*, pages 11–11, Montreal, Canada, August 2009.

[75] Steven Yantis and John Jonides. Abrupt Visual Onsets and Selective Attention: Evidence From Visual Search. *Journal of experimental psychology: Human perception and performance*, 10(5):601–621, October 1984.

[76] JoAnne Yates and Wanda Orlikowski. *The cultural turn: Communicative practices in workplaces and the professions*, chapter The PowerPoint Presentation and Its Corollaries: How Genres Shape Communicative Action in Organizations. Baywood Pub Co, 2007.

[77] Akinori Yonezawa, Jean-Pierre Briot, and Etsuya Shibayama. Object-Oriented Concurrent Programming ABCL/1. *ACM SIGPLAN Notices*, 21(11):258–268, November 1986.