2010

HTML EXENDED EXPORT TEMPLATE BUILDING GUIDE

Kushal Likhi

© Oracle

1/1/2010

OpenOffice.org

HTML EXTENDED EXPORT TEMPLATE BUILDING GUIDE

By

Kushal Likhi

This page is intentionally left blank

Table of Contents

PREFACE

This book is written by Kushal Likhi, creator of the Extended HTML Export Filter for impress. This book is a training manual which talks about the template building API and the HTML Export in detail. This manual is written with a point of view that the reader is Not a coding expert, he/she is just a layman like anybody with basic knowledge of computers and websites.

This book has 3 sections, as follows:

1) Basics(Talks about basic JavaScript knowledge)
2) The Basics and the API(Talks about how the templates work and describes the API)
3) Examples and samples

For any feedbacks and suggestions feel free to contact the author at:
Email: kushal.likhi@gmail.com
Web: http://kushal.likhi.me

Link to OpenOffice.org Project: www.openoffice.org

Link to Project Page:
http://wiki.services.openoffice.org/wiki/OpenOffice.org_Internship/Projects/2010/Customizable_html_export_for_Impress
You will find all downloads here under the, “downloads” section.

Edition: 2010
Revision: 1
©Oracle

INTRODUCTION

A new extended HTML export filter is created for OpenOffice.org (Impress), this export filter has many features including the *template support*.

These templates can define the look, feel and the User Interface for the exported HTML Presentation.

Programming Language For templates

The templates are programmed in JavaScript/CSS, as it is the scripting language supported by Browsers at the Client End.

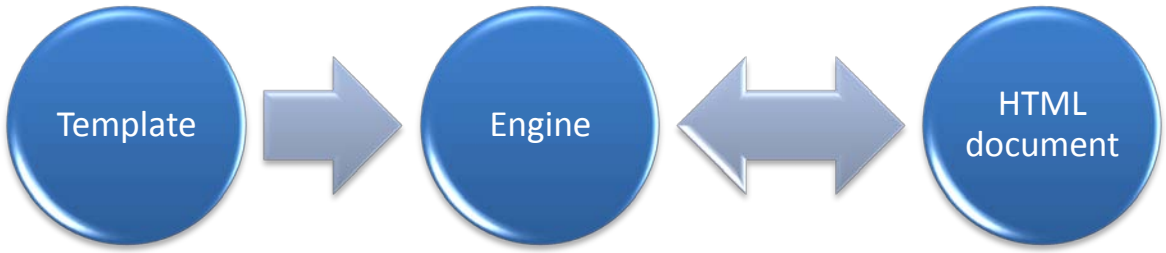
If you are a Layman, You don't need to have a in depth knowledge of the JavaScript in order to develop templates, the API library provides a very powerful set of functions, Interfaces and data in order to get the templates running within a short interval of time with all powerful features.

Whereas if you are a ambitious developer and want to develop a template with great core changes and new functionality, then the API Library also offers you all the Overrides and freedom to develop everything in depth yourself.

Brief Insight of the HTML Export Working

The exported HTML file works on the concept of NO page refreshes and links. The whole presentation runs on a single HTML files using the up to date JavaScript and DOM concepts. The elements of the HTML document are created and destroyed on runtime and all these processes are controlled by the Presentation Engine.

Templates are also interpreted by the engine and applied to the document, in other words Engine also implements Template building API and environment.



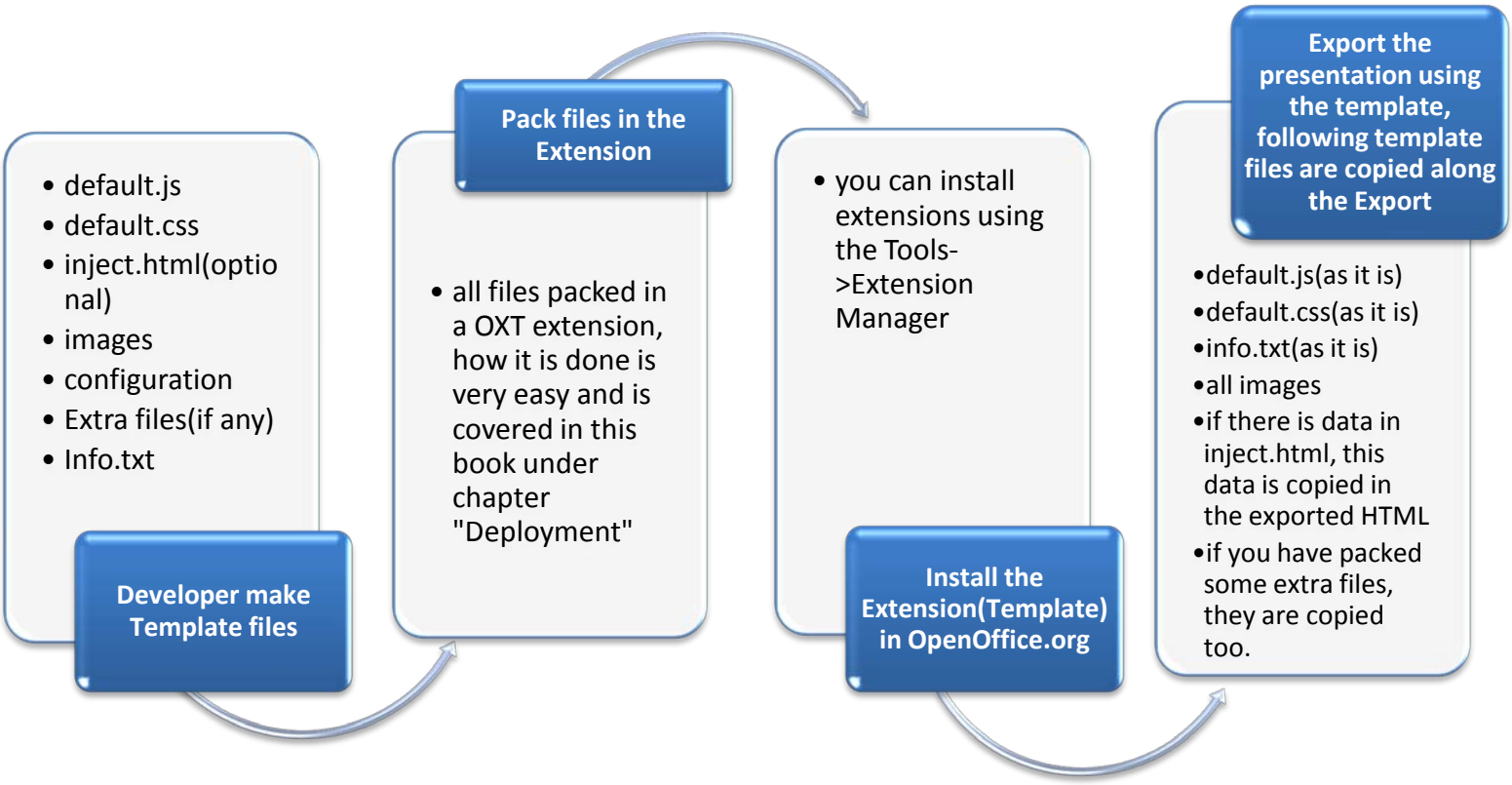
HOW TEMPLATES WORK

Templates work at the Client Browser side, but they need to be deployed as an OpenOffice.org extension, such that they can be installed and are available during the Export.

During export process the user chooses the template and the files related to the chosen template are copied with the exported HTML file. At the client browser side the **Presentation Engine** interprets these templates and makes them work.

Template Building-Deployment-Use Life Cycle

This process can be illustrated as:



Files in the Template:

Following is the list of files a template consists of following files, these files are deployed as an extension and then the export filter handles them as described:

Default.js

File type: JavaScript
How is it handled: copied as it is to the ”/template” directory of the HTML Export.
Purpose: This file defines the template dynamic layouts and functionality.

Default.css

File type: CSS
How is it handled: copied as it is to the ”/template” directory of the HTML Export.
Purpose: This file contains static CSS properties of the elements in the HTML Document.

inject.html

File type: html
How is it handled: the contents of this file are copied as it is, in the body section of the exported HTML document.
Purpose: This file is used to add some new elements to the HTML Export or some other innovative use, as required by the template. This file is optional, but required if you want to add some of your own template specific elements to the HTML document.

info.txt

File type: text file (ASCII/utf-8 encoding)
How is it handled: copied as it is to the ”/template” directory of the HTML Export.
Purpose: The information about the template and creators with the license agreement can be given here. Also tutorials for tweaking can be given here.

Images Directory

File type: Folder

How is it handled: files in this folder are copied as it is to the ”/template/images” directory of the HTML Export.

Purpose: contains all images used by the template.

Extras Directory

File type: Folder

How is it handled: files in this folder are copied as it is to the ”/template” directory of the HTML Export.

Purpose: contains extra files which could be of any format that the developer wants to export along with the other template files. These files could be some other required template files, or some disclaimers, or promotional material etc.

PREREQUISITES

Here are the things you need to know and have in order to build a template for HTML export.

Tools required:

Following tools are required:

1) A text editor to edit JavaScript/CSS/text files. I recommend Gedit as it is free and available on all platforms, whereas amongst licensed(not free) Dreamweaver is good.
2) An archive creator, WinRar is what I use in windows, others will work too, we just need to create .zip files.
3) A Browser to test the template.
4) And of-course OpenOffice to export a presentation with your template. ☺

Knowledge Required:

You need to have a basic knowledge of:

1) JavaScript
2) CSS /HTML
3) How to use WinRar.

If you **don't know JavaScript or CSS**, a small tutorial is given in the unit-1 of this book. You can also **ignore** the use of CSS/HTML and develop the template purely using JavaScript API easily, in case of very novice developers.

Other Downloads:

1) An Empty Extension template, you will find it on the project page mentioned in the Preface.(Need this for Deployment of your template)
2) This template building guide, I can presume you already have one. ☺

Part 1

PART – 1

INTRODUCTION

TO

JAVASCRIPT AND CSS

NOTE:
If you already know JavaScript and CSS then you can skip this part and directly go to Part-2

CSS BASICS

CSS is the acronym for: 'Cascading Style Sheets'. CSS is an extension to basic HTML that allows you to style your web pages.

Concept of CLASS and ID

Each element in the HTML document can be uniquely identified by using an identifier which is called an ID for that element.

We can assign ID's to any HTML Element by using "id" attribute.

Ex:

<any tag id="*identifier*">… … …</any tag>

Similarly

A group of elements in the HTML document can be identified by using a Group identifier which is known as Class.

We can assign Class to any HTML Element by using "class" attribute.

Ex:

<any tag class="*className*">… … …</any tag>

Assigning CSS Properties to a Class:

.className{
Properties here with there value;
Property 2;
}

Assigning CSS Properties to an ID:

#identifier{
Properties here with there value;
Property 2;
}

CSS Properties List:

You don't need to know much about CSS(as a starter), you just need to know about the available properties and their possible values.

The CSS properties can be categorized in following categories:

1. Background Properties
2. Border Properties
3. Classification and Positioning Properties
4. Dimension Properties
5. Font Properties
6. Generated Content Properties
7. List Properties
8. Margin Properties
9. Outline Properties
10. Padding Properties
11. Page Properties
12. Table Properties
13. Text Properties
14. Other Properties

These properties can be used to style the elements in a HTML document.

The main motive of this book is not to teach CSS, hence description and detailed list is not mentioned here. You can always Google for details, try searching "CSS property list", you will get the detailed list for reference.

This is all you need to know for getting started, and yes remember that during development you have to make all your CSS entries in default.css file.

JAVASCRIPT BASICS

This section is to be written.

Part 2

PART – 2

BASICS ABOUT TEMPLATES

and the

API Library

LAYERED ARCHITECTURE

The exported HTML document is designed to work with several layers in which elements and graphics can be placed. Few of these layers are predefined and used by the Engine, other layers are available free to use by the template designers for cutting edge GUI within the browser.

The information about the layers with their number and respective constant name

Layer No.	Description/Content	Used By Core	Free For Template	Constant Identifier
1	Cache Layer	yes	Yes	layer.cache
2	Background Mask	yes	No	layer.bg
3	Free	No	Yes	layer.bs1
4	Free	No	Yes	Layer.bs2
5	Free	No	Yes	layer.bs3
6	Active Slide	Yes	No	layer.activeSlide
7	Free	No	Yes	layer.as1
8	Free	No	Yes	layer.as2
9	Free	No	Yes	layer.as3
10	Free	No	Yes	layer.as4
11	Navigational Elements	yes	Yes	layer.navigation
12	Special Content Active	Yes	No	layer.specialContent
13	Buttons	Yes	Yes	layer.buttons
14+	Free	No	Yes	n/a

bs1-bs3 = layers below the slide(bs = below slide)
as1-as4 = layers above the displayed active slide(as = above slide)

All elements sent to cache layer will not be visible but will be a part of the HTML document. This is useful in preloading elements, increasing speed and performance and hiding unwanted objects temporarily.

How to use these layers is described later on in this book.

API LIBRARY

Template developers are provided with a library to make developing presentation templates easy. And it will also solve the problem how templates will be integrated and used.

You will find all the Library's components and call processes explained in this section.

The tutorial on implementing these components is given in Part 3 of this book.

Object Based System

The API library is defined using Objects and its members.

This is done to encapsulate the library and make it understandable and easy to use, also to finish off and chances of conflict with function or variable Names.

Objects available are described here, in detail they are discussed in the next section:

A. **Presentation:** this object is the parent object and encapsulates whole of the library.
B. **Data:** this object contains data and related tasks in relation to notes, Outline, Index and sounds.
C. **Template:** this is the object which will be used by the template to initialize and override functionalities.
D. **Client:** this object will contain client browser related tasks.
E. **Browser Info:** contains information about the browser.
F. **System:** contains info about the system.
G. **Info:** contains information about the presentation.
H. **Settings:** contains the settings for the presentation.
I. **Element:** contains element related tasks.
J. **Events**: it contains all the event handlers and overrides.
K. **SoundEngine**: contains sound related tasks.
L. **Layer**: contains info about all the layers available.

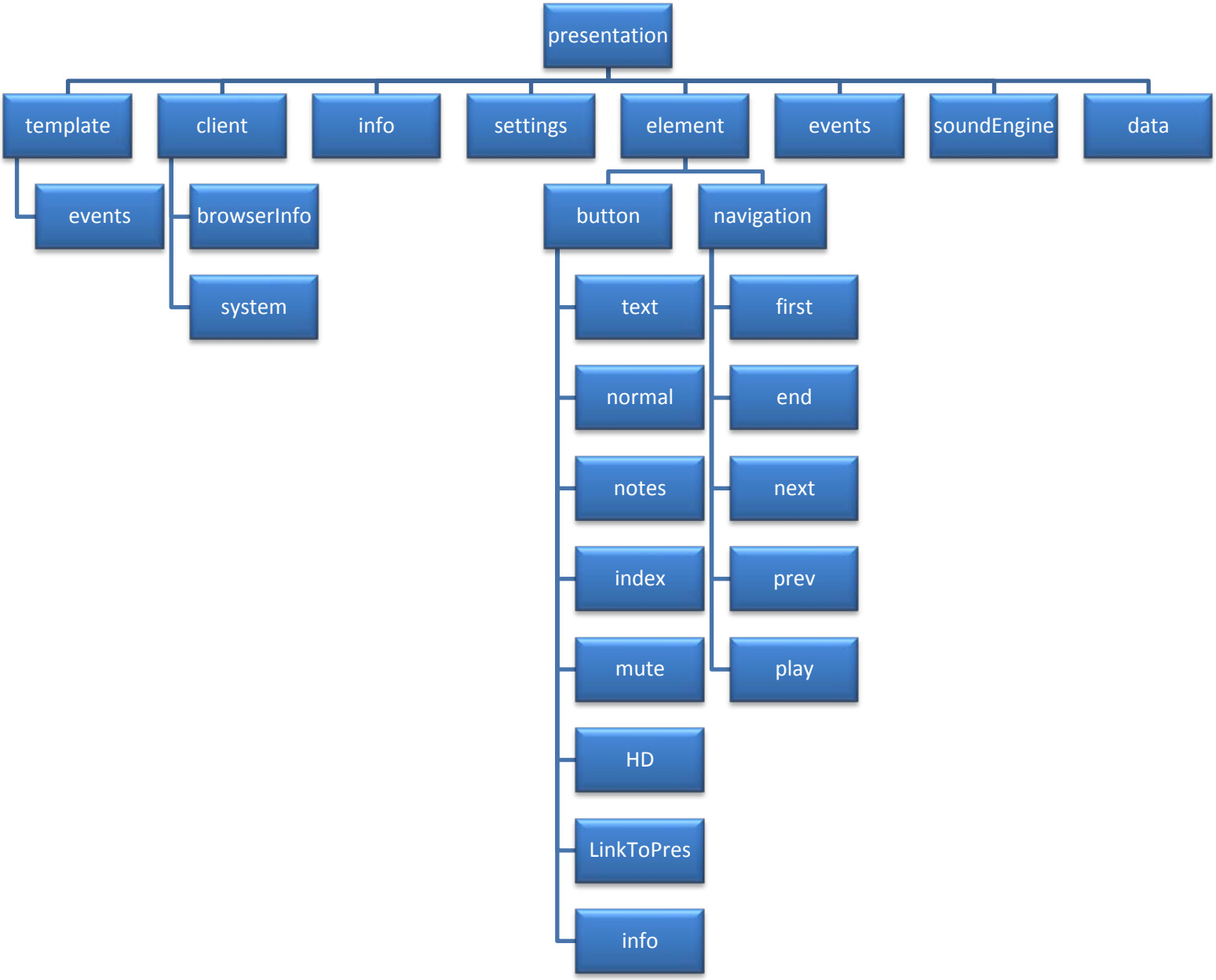
All these objects with some more are defined in a hierarchy to make itself explanatory and easy to use, and also it looks more formal and arranged ☺

The complete hierarchy is shown in the diagram below:

Understanding this chart will make you understand the library explained ahead.

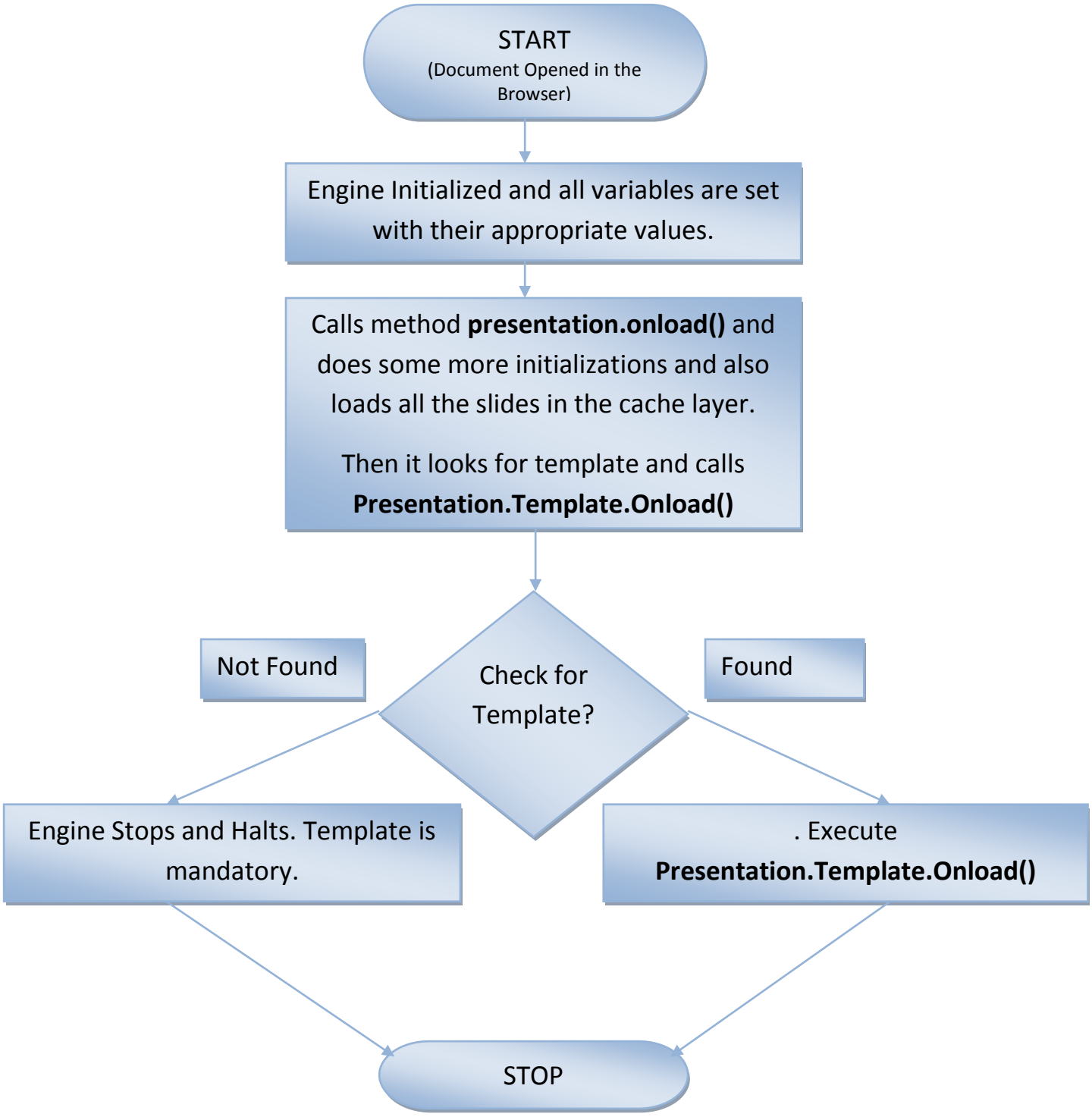
Object Hierarchy Chart

Here is the complete hierarchy chart with “presentation” object on the top encapsulating all:



The Loading Process

The loading process can be explained best using a flow chart. This flow chart describes the loading process for the HTML document and the entry point for template execution.



presentation.template.onload() (Entry Point)

Like in C/C++ we have main function as the entry point for the program, here we have this method as the entry point, it means that simply when page will load, the template will be loaded and this method will be called by the engine, making it the entry point for the template. As explained in the flow chart.

Syntax:

To define:

```
presentation.template.onload = function(){
// your code here
}
```

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.template.onload();

presentation.template.resize()

Whenever the browser window will be resized this method will be called if defined by the template. It can be used to re-place the elements according to the new window size or can be used for some other innovative use.

Syntax:

To define:

```
presentation.template.resize = function(){
// your code here
}
```

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.template.resize();

Loading Slides

Each slide is saved as an image in the /slides folder with name as sn.png where 'n' is the number of the slide.

You can write your own code for this, but engine implements a function to load the slide on the page with care to cross browser and platform support.

Generally this function is called automatically by the engine during the loading process, and user does not need to explicitly call it.

Note: slides which are loaded are not directly visible on the page, they are loaded in the cache layer of the page, this helps in preloading the slides. To show the slides use showSlide() as explained in the next section.

presentation.loadSlide(num)

This function loads a slide on to the HTML document. It creates an IMG tag and assigns it the respective id, class and image URL, then append it to the document body. The argument "num" represents the slide number to be loaded.

ID assigned is "sn", where 'n' is the slide number.

Class assigned is "slide"

Syntax:

To Call Explicitly:
This method can be explicitly called anytime by:
Presentation.loadSlide(num);

presentation.loadAllSlides()

This function loads all the slides on to the HTML document. It creates an IMG tag and assigns it the respective id, class and image URL, then append it to the document body.

ID assigned is "sn", where 'n' is the slide number.

Class assigned is "slide"

Syntax:

To Call Explicitly:
This method can be explicitly called anytime by:
Presentation.loadAllSlides();

Manipulating Elements

Following functions are used to manipulate elements on the page

presentation.element.addElement(tag,id,class)

This function is used to dynamically add an element to the document. It can add an element of any type to the Body of the HTML document at runtime.

Syntax:

To Call Explicitly:
This method can be explicitly called anytime by:
Presentation. elment.addElemente(tag,id,class);

Parameters:
Tag = a string containing the tag you want to add. Ex. “img”
Id = a string which contains the ID you want to assign to that element
Class = a string which contains the ClassName you want to assign to this element.

presentation.element.removeElementById(id)

This function is used to remove an element from the document. It can remove an element of any type from the HTML document at runtime.

Syntax:

To Call Explicitly:
This method can be explicitly called anytime by:
presentation.element.removeElementById(id)
Parameters:
Id = a string which contains the ID of the element you want to remove

presentation.element.addCssJs(filename, filetype)

This function can assign the external CSS or JS files to the document at runtime.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.element.addCssJs(filename, filetype)

Parameters:

filename = a string which contains the path to the file.

filetype = a string which either “css” or “js” which specifies the type of file.

presentation.element.place(id,top,left)

This function is used to place an element on the HTML document at the appropriate position. This function uses the **percentage system**, i.e. the parameters top/left are not in absolute pixels value, they are in percentage with reference to the total height/width. This is very useful in maintaining the position of the element on the screen on all various different resolution and platforms. In other words the element is placed relative to the screen and hence is always at the desired position, which is not true in the case of absolute positioning. Engine automatically does all the calculations and places the element.

Ex. Suppose you have the screen area of 100x200 Pixels, and you observed that in your monitor the element looks best when placed at top=20, left=20 in absolute pixels value, then the parameters passed in percentage to the function will be
top=((20/100)*100) = 20% (used the simple Percentage formula)
left=((20/200)*100) = 10% (used the simple Percentage formula)
and now on any screen with any resolution the elements will be placed appropriately on the desired location.

Note: *you don’t need all these formulas and calculations, it was just to explain, while building templates, from the start just experiment with the percentage values to get the right position.* ☺

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.element.place(id,top,left)

Parameters:

id = a string which contains the ID of the element to be placed.

top = a float which contains the numeric “top“ value in percentage of the top left corner of the element.

left = a float which contains the numeric “left “ value in percentage of the top left corner of the element.

presentation.element.size(id,height,width)

This function is used to scale an element on the HTML document to the appropriate size. This function uses the **percentage system**, i.e. the parameters height/width are not in absolute pixels value, they are in percentage with reference to the total height/width. This is very useful in maintaining the size of the element on the screen on all various different resolution and platforms. In other words the element is scaled relative to the screen and hence is always of the desired size, which is not true in the case of fixed absolute size. Engine automatically does all the calculations and scales the element.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.element.size(id,height,width)

Parameters:

id = a string which contains the ID of the element to be placed.

width = an int which contains the numeric “width“ in percentage of the element.

height = an int which contains the numeric “height“ in percentage of the element.

presentation.element. placeNsize(id,top,left,height,width)

This function is used to scale and place the element in the same time. i.e. contains the combined functionality of **presentation.element.size** and **presentation.element.place** , it also follows the **percentage system** as explained before.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.element. placeNsize(id,top,left,height,width)

Parameters:

id = a string which contains the ID of the element to be placed and scaled.
width = an int which contains the numeric "width" in percentage of the element.
height = an int which contains the numeric "height" in percentage of the element.
top = a float which contains the numeric "top" value in percentage of the top left corner of the element.
left = a float which contains the numeric "left " value in percentage of the top left corner of the element.

presentation.element. placeNsizeByClass(class,top,left,height,width)

This function is used to scale and place all the elements with the **class name** specified. It contains the combined functionality of **presentation.element.size** and **presentation.element.place** , it also follows the **percentage system** as explained before.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.element. placeNsizeByClass(class,top,left,height,width)

Parameters:

class = a string which contains the Class name of the elements to be placed and scaled.
width = an int which contains the numeric "width" in percentage of the element.

height = an int which contains the numeric “height“ in percentage of the element.
top = a float which contains the numeric “top“ value in percentage of the top left corner of the element.
left = a float which contains the numeric “left “ value in percentage of the top left corner of the element.

presentation.element. aplace(id,top,left)

This function is used to place an element at the absolute position on the screen.

The parameter values top/left are absolute pixel value.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.element.aplace(id,top,left)

Parameters:

id = a string which contains the ID of the element to be placed and scaled.
top = an int which contains the numeric “top“ value in pixels of the top left corner of the element.
left = an int which contains the numeric “left“ value in pixels of the top left corner of the element.

presentation.element.asize(id, height,width)

This function is used to size an element on the screen.

The parameter values width/height are the absolute pixel value.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.element.asize(id,height,width)

Parameters:

id = a string which contains the ID of the element to be scaled.
Height = height of the element in pixels.

Width = width of the elements in pixels.

presentation.element. aplaceNsize(id,top,left,height,width)

This function is used to place and scale an element on the screen.

The parameter values top/left/height/width are absolute pixel value.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.element.aplaceNsize(id,top,left,height,width)

Parameters:

id = a string which contains the ID of the element to be placed and scaled.

top = an int which contains the numeric “top“ value in pixels of the top left corner of the element.

left = an int which contains the numeric “left“ value in pixels of the top left corner of the element.

Height = height of the element in pixels.

Width = width of the elements in pixels.

Data and State Variables Available

This is the list of all data variables and state variables which are available for use.

S.No		Type	Description
1	presentation.info.currentSlide	Int	Contains the serial number of the slide currently displayed.
2	presentation.client.width	int	Contains page width in pixels
3	presentation.client.height	Int	Contains page height in pixels
4	presentation.info.author	String	Contains the name of the author of the document
5	presentation.info.slideCount	Int	Contains total number of slides available in the presentation
6	presentation.info.title	String	Stores the title of the presentation
7	presentation.info.version	String	Contains the version of the OpenOffice.org used to export this presentation
8	presentation.info.extensionVersion	String	The version of the HTML Export extension used for the export.
9	presentation.info.timeStamp	String	The date and time of the export
10	presentation.settings.outline	Bool	Set true if outline view is enabled
11	presentation.settings.notes	Bool	Set true if notes view is available

12	presentation.settings.index	Bool	Set true is index is available
13	presentation.settings.sound	Bool	Set true if sounds are available
14	presentation.settings.indexOnLoad	Bool	Set true if the index is supposed to be shown at the start.
15	presentation.settings.autoTime	Int	Time in milliseconds after which slides have to be automatically changed in the play mode.
16	presentation.settings.webcastStatus	Int	Non zero value if webcast is available.
17	presentation.settings.playMode	Bool	This is a flag which is set true if the auto play mode is on
18	presentation.settings.notesMode	Bool	This flag is true if notes are currently displayed
19	presentation.settings.soundMute	Bool	This is a flag which is true if sounds are switched on.
20	presentation.settings.outlineMode	Bool	This flag is true for the period outline of the document is displayed
21	presentation.settings.indexMode	Bool	This flag is true for the period index is displayed.
22	presentation.client.browserInfo.name	String	Contains the name of the browser in which document is opened
23	presentation.client.browserInfo.codename	String	Contains codename of the browser

24	presentation.client.browserInfo.version	String	Contains the version of the browser
25	presentation.client.browserInfo.platform	String	Contains the platform of the client browser
26	presentation.client.browserInfo.javaEnabled	Bool	Set true if JavaScript is enabled
27	presentation.client.system.platform	String	Contains the system platform

Apart from the above data and state variables, we also have data for various different views made easily available to you.

Getting Index Data

The index details are made available to the template developer, and the template developer can show this data in any way as desired.

This data is saved in the presentation.data Object.

The index and page title for the slides are extracted and stored in an array by the engine, i.e.

Presentation.data.indexArray

The length of this array is equal to the number of slides in the presentation which is stored in the variable “presentation.info.slideCount”

To extract the index data for a particular slide, there is a method which is very handy and should be used for flawless operation, i.e.

Presentation.data.indexData(slideNo)

This function will return the string containing the page title for the slide number passed as the argument.

ex: to get the index page title of the slide number 1 you can give command

var PageTitle = presentation.data.indexData(1);

now PageTitle variable will have a string containing title for the current slide.

Getting Notes View Data

A slide can have notes associated to it, hence the notes for each slide are made available to the template developer, and the template developer can show this data in any way as desired.

This data is saved in the presentation.data Object.

The notes for the slides are extracted and stored in an array by the engine, i.e.

Presentation.data.notesArray

The length of this array is equal to the number of slides in the presentation which is stored in the variable “presentation.info.slideCount”

To extract the notes data for a particular slide, there is a method which is very handy and should be used for flawless operation, i.e.

Presentation.data.notesData(slideNo)

This function will return the string containing notes for the slide number passed as the argument.

ex: to get the notes of the current slide you can give command

var Notes = presentation.data.indexData(presentation.info.currentSlide);

now notes variable will have a string containing notes for the current slide.

Getting Sound file Data

A slide can have sounds associated to it, which should be played on slide transition. hence the sound files for each slide are made available to the template

developer, the template developer can use inbuilt sound engine to handle them or can use them innovatively.

This data is saved in the “presentation.data” Object.

The associated sounds for the slides are extracted and stored in an array by the engine, i.e.

Presentation.data.soundArray

The length of this array is equal to the number of slides in the presentation which is stored in the variable “presentation.info.slideCount”

To extract the sound file for a particular slide, there is a method which is very handy and should be used for flawless operation, i.e.

Presentation.data.soundFile(slideNo)

This function will return the string containing the path to the sound file for the slide number passed as the argument.

ex: to get the sound file for the current slide you can give command

```
var soundFile = presentation.data.soundFile(presentation.info.currentSlide);
```

Now soundFile variable will have a string containing path to sound file for the current slide.

Getting Outline/Text View Data

All the slides are also available in Text format. It is for providing the plain text for visually impaired personals or for other purposes.

Data is divided in three parts and stored in the arrays. The three parts are “Slide title/heading”, “subtitle” and “body”. The names are self explanatory about their contents.

The arrays in which this data is stored are:
presentation.data.outlineTitleArray
presentation.data.outlineSubtitleArray

presentation.data.outlineTextArray = new Array();

To extract the Outline/Text view data for a particular slide, following methods could be used

Presentation.data.OutlineTitle (slideNo)

Presentation.data.OutlineSubtitle (slideNo)

Presentation.data.OutlineText(slideNo)

Above function will return the corresponding strings containing the Outline/Text for the slide number passed as the argument.

Handling Audio using inbuilt Audio Engine

The audio support is provided to the template developers through very simple methods implemented by the sound engine.

The sound engine uses the HTML 5 Audio features and hence is compatible with all HTML 5 browsers (i.e. all latest browsers), I recommend using .wav file format as it is compatible with most browsers, others like .ogg and .mp3 are not that popular with the browsers.

The sound API is divided in simple parts, i.e.

1) Handling volume
2) Playing default audio
3) Dynamic sound objects

Each part is explained as follows:

Handling Volume

Volume is stored in the variable ***"presentation.soundEngine.volume"***. you can read this variable to monitor current sound level(0-100)

To set the default volume or change the volume use the following function:

----presentation.soundEngine.setVolume(volume)

The parameter “Volume” is an integer value between 0-100

It will set the volume to the specified value.

Please pass the values within the range of 0-100 only as an integer.

Ex. presentation.soundEngine.setVolume(90);

As an usage example, Template developer can also provide a volume control on the screen and call this function to change the volume real time whenever that control’s state is changed.

Playing Default Audio

Each slide could have sound associated with it which could be played by calling this function:

----presentation.soundEngine.play(SlideNo)

When this function is called then it will play the sound associated with the slide whose number is passed as an argument.

It will automatically look for the associated object, create its object and play it all encapsulated in this single function to give the maximum ease of use.

Ex: presentation.soundEngine.play(presentation.info.currentSlide);

The argument “SlideNo” is of type integer and contains the number of the slide whose associated audio has to be played.

Dynamic Sound Objects

Consider a case where the template developer wants to play some custom sounds packed at some event, or even the sounds packed with the slides, then for this purpose the developer can create real time sound objects and use them to play sounds.

To create a sound object from the source URL use the function:

----presentation.soundEngine.getAudioObject(URL)

This function will accept the URL of the audio file and return the audio object.

This audio object has following methods to control it

1) play: it will play the audio.
2) pause: it will pause the playback.
3) volume: it will set the volume of that particular audio stream. You can have several objects played with different volume at the same time. The volume will be passed as an “float” with values between 0-1.

Ex:

Lets create some music using the sound objects:

```
Var drum = presentation.soundEngine.getAudioObject(“drums/drum.wav”);
Var guitar = presentation.soundEngine.getAudioObject(“guitar.wav”);
Var bass = presentation.soundEngine.getAudioObject(“basses/bass2.ogg”);
drum.volume = 0.60;
guitar.volume = 0.91;
bass.volume = 0.88;
guitar.play();
bass.play();
drum.play();
```

Miscellaneous Functions

Other functions are listed here:

presentation.client.getSize

This function determines the available height and width for the document in the browser window in pixels. In other words it finds the height and width of the white space in the browser window where webpage is displayed.

Determined height and width is stored in the global variables:

"**presentation.client.width**" and "**presentation.client.height**"

Syntax:

To Call Explicitly:
This method can be explicitly called anytime by:
Presentation.client.getSize();

presentation.element.sendToLayer(string eid, String Lid)

This function sends the element with ID passed as eid(element ID), to the layer passed as 'Lid'(see layer Object).

Ex: presentation.element.sendToLayer("id",layer.cache);

Element could be of any type.

It is used to shift the elements between several layers.

Syntax:

To Call Explicitly:
This method can be explicitly called anytime by:
Presentation.element.sendToLayer(eid, Lid);

presentation.printPage()

This function will print the page.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

Presentation.printPage();

presentation.readCookie(name)

This function will read a cookie and **return** its value.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.readcookie(name)

Parameters:

Name = the name of the cookie to be read.

presentation.writeCookie(name,value,time)

This function will create a cookie using the supplied parameters.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.writeCookie(name,value,time)

Parameters:

Name = Name of the cookie

Value = Value to be stored in this cookie

Time = Time in Hours for which the cookie is to be available

presentation.preloadImage(<args list>)

This function will the images. URL's of the images is passed as the arguments list.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

presentation.preloadImage(<img1>,<img2>,<img3>, .. ,<img n>)

Parameters:

'N' number of parameters each containing the string, path/URL to an image.

presentation.client.statusMsg(message)

This function will write a message passed as a string to the status bar of the client browser.

Syntax:

To Call Explicitly:

This method can be explicitly called anytime by:

Presentation.client.statusMsg(message)

Parameters:

Message = string which is to be shown at the status bar.

Working with Buttons

Template developer can have any number of buttons on the screen but few useful and required buttons are defined in the engine and can be easily implemented using simple functions. Their functionality is also internally defined hence the template developer does not have to spend time in implementing them. But on the other side if the template developer does not want to use internal functionality, he/she can override the internal functionality to define its own, how this is done is explained in the next section(events).

These buttons don't have any predefined shape or look, they are truly transparent, their size, image, position and other visual properties are defined by the template developer. This is done to make the buttons more generic and to be able to fit in all implementations.

It is not mandatory to use these buttons, you can define your own buttons.

Buttons available are:

1) Navigation
    a. **Start:** go to the start of the presentation(slide 1)
    b. **End:** go to the last file directly.
    c. **Prev:** to go to the previous slide.
    d. **Next:** to go to the next slide.
    e. **Play:** play the auto increment system. Slides will change(progress) after each fragment of time. The fragment of time is defined in "auto Time"
2) Others
    a. Info: will display the presentation info
    b. **Link To Presentation:** this button will be a link to downloading the presentation in ODF format.
    c. **HD:** this button will activate HD viewing.
    d. **Index:** will display the index
    e. **Notes:** will display the notes on the screen.
    f. **Mute:** will mute the sound.

g. **Normal:** will show the slide, hiding all other views.
h. **Text:** this will show the outline text of the document.

Each of these buttons have following properties and methods associated with them to make it easy and simple to use

1) Properties: Two properties:
    a. **ID:** it contains the Identifier for the button, can be used to manipulate the button. By default the ID for the button is same as its name, ex: next, back etc. but if you want to assign your own custom ID, then set the ID property before you call the Add() function.
    b. **Class:** contains the class the button belongs to. It can be used in the same way as the ID is used.
2) Methods: Two of the methods:
    a. **Add(image):** this method will add the button on the document, the parameter Image is the URL to the image file for the button.
    b. **Remove():** It will remove the button from the presentation page.
3) Event Handlers: there are 3 event handlers assigned to each button:
    a. **OnClick:** this is executed when we click on the button. There is an inbuilt functionality to handle these events, hence you don't have to worry about it, but still if you want to add your own handlers and functionality there are overrides available. Its discussed in detail in the section "handling events"
    b. **OnMouseOver:** it is executed when mouse pointer is brought over the button. How to use it in detail is discussed in the section "handling events".
    c. **OnMouseOut:** this event is triggered when the mouse pointer goes away from the bounds of the button. How to use it in detail is discussed in the section "handling events".

ID's

The ID's for each button are defined in the table below:

S.No	Variable containing "ID"	Button Type	Default Value
1	presentation.element.navigation.first.id	Start	FirstButton
2	presentation.element.navigation.end.id	End	EndButton
3	presentation.element.navigation.next.id	Next	NextButton
4	presentation.element.navigation.prev.id	Previous	PrevButton
5	presentation.element.navigation.play.id	Play	PlayButton
6	presentation.element.button.notes.id	Notes	NotesButton
7	presentation.element.button.text.id	Outline	TextButton
8	presentation.element.button.mute.id	Mute	MuteButton
9	presentation.element.button.normal.id	Normal	NormalButton
10	presentation.element.button.index.id	Index	IndexButton
11	presentation.element.button.linkToPres.id	Link to Pres	L2PButton
12	presentation.element.button.hd.id	HD	HDButton
13	presentation.element.button.info.id	Info	InfoButton

Classes:

Similar to the ID, each button belongs to a class, it is stored in a variable which is mentioned below:

By default there are two classes i.e. "navigation" and "button", navigation class refers to the navigational Buttons and "button" class refers to the other buttons,, its shown in the table below:

S.No	Variable containing "Class"	Button Type	Default Value
1	presentation.element.navigation.first.eclass	Start	navigation
2	presentation.element.navigation.end. eclass	End	navigation
3	presentation.element.navigation.next. eclass	Next	navigation
4	presentation.element.navigation.prev. eclass	Previous	navigation
5	presentation.element.navigation.play. eclass	Play	navigation
6	presentation.element.button.notes. eclass	Notes	button
7	presentation.element.button.text. eclass	Outline	button
8	presentation.element.button.mute. eclass	Mute	button
9	presentation.element.button.normal. eclass	Normal	button
10	presentation.element.button.index. eclass	Index	button
11	presentation.element.button.linkToPres. eclass	Link to Pres	button
12	presentation.element.button.hd. eclass	HD	button
13	presentation.element.button.info. eclass	Info	button

ADD(imageURL) – Adding the button on the document

You can call add(image) function to add the corresponding button to the document.

This function will have all the default functionality and event handlers assigned automatically to the image specified as an argument, hence making that image an image button. You can specify styling for that image using the ID for that button. ID can be retrieved as shown above.

imageURL(argument): link to the image you want to display as the button. Images can be on local or global servers.

Ex:

Presentation.navigation.next.add("images/nextButton.png");

Button specific call to the add() method is given below

S. No	Variable containing "Class"	Button Type	Default Value
1	presentation.element.navigation.first.add(i)	Start	navigation
2	presentation.element.navigation.end. add(i)	End	navigation
3	presentation.element.navigation.next. add(i)	Next	navigation
4	presentation.element.navigation.prev. add(i)	Previous	navigation
5	presentation.element.navigation.play. add(i)	Play	navigation
6	presentation.element.button.notes. add(i)	Notes	button
7	presentation.element.button.text. add(i)	Outline	button
8	presentation.element.button.mute. add(i)	Mute	button
9	presentation.element.button.normal. add(i)	Normal	button
10	presentation.element.button.index. add(i)	Index	button
11	presentation.element.button.linkToPres.	Link to	button

	add(i)	Pres	
12	presentation.element.button.hd. add(i)	HD	button
13	presentation.element.button.info. add(i)	Info	button

*Parameter 'I' in the table above represents the "ImageURL"

Remove() – Removing the button from the document

You can remove the button anytime from the document by just calling the remove() method.

Ex:

Presentation.navigation.next.remove();

Button specific call to the remove() method is given below

S. No	Variable containing "Class"	Button Type	Default Value
1	presentation.element.navigation.first.remove( )	Start	navigation
2	presentation.element.navigation.end. remove()	End	navigation
3	presentation.element.navigation.next. remove()	Next	navigation
4	presentation.element.navigation.prev. remove()	Previou s	navigation
5	presentation.element.navigation.play. remove()	Play	navigation
6	presentation.element.button.notes. remove()	Notes	button
7	presentation.element.button.text. remove()	Outline	button
8	presentation.element.button.mute. remove()	Mute	button

9	presentation.element.button.normal. remove()	Normal	button
10	presentation.element.button.index. remove()	Index	button
11	presentation.element.button.linkToPres. remove()	Link to Pres	button
12	presentation.element.button.hd. remove()	HD	button
13	presentation.element.button.info. remove()	Info	button

Handling Events

Part to be written

Part 3

PART – 3

TUTORIALS ON TEMPLATE BUILDING

And application Examples Of

API Library

Section to be written

Part 4

PART – 4

DEPLOYING TEMPLATES

DEPLOYING TEMPLATE AS EXTENSION

The steps can be found at

http://wiki.services.openoffice.org/wiki/OpenOffice.org_Internship/Projects/2010/Customizable_html_export_for_Impress#Steps_to_build_your_own_add-on_extension