**PPTX Translation to HTML**

1. **Overview and Links to Relevant Documentation**
   1. Introduction
   2. Links to Relevant Documentation
2. **The PowerPoint Format**
   1. Inspecting the XML Contents of a PPTX File
   2. Shape Property Inheritance
   3. English Metric Units (EMU)
3. **Imported Python Libraries**
   1. html-1.16
   2. python-pptx
4. **pptx\_to\_html Code**
   1. Overview
   2. Output Format: SVG
   3. Output Scaling and Coordinate Systems
   4. shapedraw.py and Other Modules
5. **Extending Functionality of python-pptx Library**

**\*1. Overview and Links to Relevant Documentation\***

**Introduction**

The following document details the implementation of **pptx\_to\_html**, a program that translates files in the Microsoft PPTX format to HTML.

**pptx\_to\_html** is written in Python. It reads in pptx files using the python-pptx PowerPoint parsing library, processes each slide individually, and for each shape found in the pptx file, generates a corresponding svg shape in the result html file.

To generate the HTML code, **pptx\_to\_html** uses html-1.16, a python library for generation of html.

**Links to Relevant Documentation**

Documentation for the python-pptx library can be found at [http://python-pptx.readthedocs.io/en/latest/#](http://python-pptx.readthedocs.io/en/latest/), and the source code for the library can be found at <https://github.com/scanny/python-pptx>.

Documentation for the html-1.16 library can be found at <https://pypi.python.org/pypi/html>.

Documentation for the Microsoft Open XML, of which the PPTX XML format is a subset, can be found at <http://www.officeopenxml.com/anatomyofOOXML-pptx.php>. This is a very useful resource when inspecting the actual XML contents of a pptx file. A full Markup Language Reference for the Office Open XML file formats, as well as documentation regarding how an OOXML package is put together can be found at <http://www.ecma-international.org/publications/standards/Ecma-376.htm>.

For instructions to using **pptx\_to\_html**, please consult the README.txt file included with the **pptx\_to\_html** program.

**\*2. The PowerPoint Format\***

**Inspecting the XML Contents of a PPTX File**

In order to understand how **pptx\_to\_html** works, it is necessary to first have an understanding of the pptx format. For a more detailed description of elements of the PowerPointML format, see <http://www.officeopenxml.com/anatomyofOOXML-pptx.php>, as well as the Markup Language reference downloadable at <http://www.ecma-international.org/publications/standards/Ecma-376.htm>. The 1st Edition Part 4 (Markup Language Reference) was one of my primary references.

To inspect the XML contents of a pptx file, copy the file and rename it with a .zip extension. Decompressing the .zip file will yield a folder containing all of the XML parts that make up the pptx file.

At the most basic level, a PowerPoint presentation is a collection of slides. Furthermore, each slide contains content in the form of a collection of shapes.

The XML for each slide can be found in the /ppt/slides folder in the unzipped pptx folder.

A shape is represented as a <p:sp> element in the slide XML. Each shape has its own set of properties, such as x,y position on its slide, bounding box size, shape type (rectangle, ellipse, line, etc.), font and formatting information for text within the shape (if it contains text), and line and fill properties, among others.

**Shape Property Inheritance**

Shape properties can be explicitly set and thus found as attributes or children of the <p:sp> element itself, but they can also be implicitly set and inherited from the other parts of the pptx file, namely a Slide Layout (the part for which can found in /ppt/slideLayouts in the unzipped pptx folder), a Slide Master (found in /ppt/slideMasters), or the document Theme (found in /ppt/theme).

Shapes come in two main flavors: placeholders versus non-placeholders. A shape is a placeholder if it contains a <p:ph> element in its <p:nvPr> (nonviewable properties) child element. If a shape is a placeholder, then it is possible for that shape to inherit properties from its Slide Layout and Slide Master.

Each slides inherits its placeholders from a Slide Layout. One can see which Slide Layout a certain slide inherits from by inspecting the relationship file found in the /ppt/slides/\_rels folder that corresponds to the particular slide.

Each Slide Layout inherits from a Slide Master. Although it is possible to have multiple Slide Masters in a pptx file, it is generally the case that only one is used. To see which Slide Master a particular Slide Layout is inheriting from, one can inspect the relationship file found in the /ppt/slideLayouts/\_rels folder that corresponds to the slide.

Thus the inheritance hierarchy is as follows: if a particular property of a placeholder shape is not specified at the shape level, i.e. the property does not show up as an attribute or child of the shape’s <p:sp> element, then we look first at the shape’s corresponding placeholder on the corresponding Slide Layout. If the property is still not specified at the Slide Layout level, then we look to the Slide Master corresponding to the Slide Layout and see if the property is specified there.

**English Metric Units (EMU)**

Within the PowerPointML format, lengths are generally stored as English Metric Units (EMU). For reference, the conversion rate is 1 inch = 914400 EMU.

Although most lengths are stored in EMUs, there are a few exceptions to this convention, the most common one being that text font sizes are stored as centipoints. Thus a 12 point font corresponds to an sz attribute in the format of 1200. The conversion rate for points is 1 inch = 72 points = 7200 centipoints.

The python-pptx library that I used to parse my PowerPoint files contains some useful functions in its pptx.util module that facilitate the easy conversion between these units. Usage of pptx.util is covered in Section 3.

**\*3. Imported Python Libraries\***

**html-1.16**

In order to generate html output, I used the html-1.16 python library. Core this library is the html.HTML object, which represents an HTML node or tree. We build our output html file by initializing a root html.HTML node, and then adding new nodes for slides and shapes. A detailed guide to using html-1.16 can be found at <https://pypi.python.org/pypi/html>.

**python-pptx**

HOW PYTHON-PPTX WORKS

The python-pptx library is used to parse pptx files into pptx.Presentation objects.

The python-pptx parser uses the lxml XML parser to parse the XML parts of a pptx file and stores them as lxml.etree.Element objects. Documentation for the lxml parser can be found at <lxml.de>.

The object classes in the python-pptx library have essentially a one-to-one correspondence with the elements found in the PPTX XML. Each pptx.Presentation object contains a list of pptx.Slide objects, which represent individual slides on the PowerPoint, and each pptx.Slide object contains tree of pptx.Shape objects, which represent the slide’s individual shapes (<p:sp> shape elements). Furthermore each pptx.Shape contains a number of attributes corresponding to the shape’s actual XML attributes and children, and these attributes are accessed and used to generate html translations of the shapes.

Detailed documentation for the python-pptx library can be found at [http://python-pptx.readthedocs.io/en/latest/#](http://python-pptx.readthedocs.io/en/latest/), and the source code for the library can be found at <https://github.com/scanny/python-pptx>.

INADEQUACIES OF THE PYTHON-PPTX LIBRARY

In many instances, I found that the python-pptx parser was inadequate for my needs, e.g. it was unable to parse certain shape properties, or certain features, such as parsing of shape groups, had not yet been implemented. Fortunately, all instances of python-pptx classes that correspond to elements in the PPTX XML contain references to the lxml.etree.Element objects from which they were created (usually as an “element” or “\_element” attribute of the instance). Thus in cases where the python-pptx parser fell short, I was able to essentially inspect the actual XML for the information that I needed. Further details are on the XML inspection process are covered in Section 5.

LENGTH CONVERSIONS AND PPTX.UTIL

To easily deal with the different length units that the PowerPointML format uses, the python-pptx library contains a library with the name pptx.util, which contains several classes and methods for converting between length units. Detailed documentation of this module can be found at <https://python-pptx.readthedocs.io/en/latest/user/autoshapes.html> under the section entitled “Understanding English Metric Units” and at <https://python-pptx.readthedocs.io/en/latest/api/util.html#util>.

**\*4. pptx\_to\_html Code\***

**Overview**

The pptx\_to\_html program consists of two main files: pptx\_to\_html.py, which handles IO actions such as reading in command line arguments, opening the correct PPTX file, and outputting to the correct HTML files; and shapedraw.py, which handles the translation of PPTX shapes into SVG nodes.

shapedraw.py imports a number of helper functions from shapetext.py, where text and text formatting parsing and rendering is implemented; shapecolor.py, where shape color parsing and rendering is implemented; and shapeutil.py, in which various miscellaneous useful helper functions are implemented.

shapedraw.py defines one primary method called draw\_shape which takes as input one PowerPoint shape (a pptx.Shape object) and translates that shape into HTML (specifically an SVG element).

pptx\_to\_html.py works by opening a designated pptx file, parsing it with the python-pptx library, iterating through each of the parsed file’s pptx.Slide objects, and calling the draw\_shape method on each of the slide’s pptx.Shape objects.

**Output Format: SVG**

pptx\_to\_html translates PPTX shapes into SVG (Scalable Vector Graphic) elements. The output HTML file is composed of these SVG elements.

The root node of the output HTML contains one root SVG node per slide that is being drawn. Each of these slide SVG nodes has height and width equal to [slide height in inches]\*PPI pixels and [slide width in inches]\*PPI pixels, respectively. (PPI, or Pixels Per Inch, is a value specified in the pptx\_to\_html code which determines how many pixels one inch should correspond to in the output HTML. Documentation on PPI can be found in the next subsection “Output Scaling and Coordinate Systems”). The slide SVG’s viewbox, which can be seen as the SVG’s local coordinate system, is defined with top left corner (0, 0) and width and height equal to the number of width pixels and height pixels, respectively.

For now, PPTX Rectangles are drawn as SVG Rect nodes; PPTX Ovals are drawn as SVG Ellipse nodes; PPTX Straight Connectors are drawn as SVG Line nodes; and all other shapes are rendered as their bounding boxes.

pptx\_to\_html renders text using SVG ForeignObject elements. Text is rendered in these foreignObjects using standard HTML elements.

**Output Scaling and Coordinate Systems**

In order to control how large of an HTML output he wants, the user can specify a value known as PPI, which stands for pixels per inch. Currently in pptx\_to\_html.py, PPI is hard coded as 50 for standard usage and 75 for slideshow mode. This means that for standard usage, an inch in the original pptx file will be rendered with the length of 50 pixels.

As mentioned in the previous subsection, PPTX slides are drawn as SVG elements in the output HTML file. Each slide corresponds to an SVG child element of the root HTML node. Each slide SVG has actual width and height equal to width\_px = [slide width in inches]\*PPI pixels and height\_px = [slide height in inches]\*PPI pixels, respectively. The viewboxes for the slide SVG elements is set at “0 0 width\_px height\_px”. This means that the SVG’s local coordinate system has a 1-1 correspondence to actual locations in the SVG, relative to the SVG’s top left corner. For example, the value (12, 37) in the SVG’s local coordinate system refers to the location 12 pixels to the right of and 37 pixels down from the SVG’s top left corner (which is (0, 0)).

It must be noted that locational and length values in the pptx\_to\_html code are all stored and calculated in terms of their EMU values (except for certain cases like text font sizes). These EMU values are converted to pixel values (using the emu\_to\_px and str\_emu\_to\_px functions found in shapeutil.py) directly prior to being put into the output HTML.

**shapedraw.py**

As stated before, the shapedraw.py file’s main purpose is to implement the draw\_shape method. The draw\_shape method takes seven arguments:

parent\_html - The parent HTML node where we would like to draw the shape. In pptx\_to\_html.py, we pass the html node of the slide as the parent html.

pres - A reference to the pptx.Presentation object the shape is contained in.

slide - A reference to the shape’s corresponding pptx.Slide object.

shape - A reference to the pptx.Shape object to be drawn.

ppi - An integer pixels per inch value.

off\_x, off\_y - Two integer values (in EMU) that determine which location the draw\_shape method will reference as the origin. If no off\_x or off\_y values are passed, off\_x and off\_y both default to 0. These values are used to handle Group shape translations.

scl\_x, scl\_y – Two float values which determine the scaling factor of the shape to be drawn. If no scl\_x or scl\_y values are passed, they default to 1.0. These values are used to handle Group shape scalings.

The pres, slide, and shape arguments provide access to the inheritance hierarchy of shape properties.

OVERVIEW OF MODULES

shapedraw.py deals mainly with the rendering of the rendering of shapes themselves, and thus implements a number of helper functions to aid in handling the geometries involved with shape transformations. shapedraw.py also implements the getters for shape line and fill styles.

shapetext.py deals with the rendering of PowerPoint text. This module implements a number of shape text style and formatting getters. The most important function to note is the draw\_text function, which draws shape textframes, outputting text in the correct locations and applying the correct stylings and formattings.

shapecolor.py deals with the parsing of color information. In addition to handling color inheritance, shapecolor.py also handles the mapping of PPTX theme colors to actual RGB values.

shapeutil.py contains useful helpful functions that didn’t really fit into any of the above categories of helper functions. Currently shapeutil.py implements the emu-to-pixel conversion function, as well as function that replaces spaces in text with '&emsp14' for outputting to HTML.

**\*5. Extending Functionality of python-pptx Parser\***

As noted in Section 3, due to the fact that the python-pptx library is still being developed, there are a number PowerPoint elements and properties that the library cannot currently parse.

We can often remedy these insufficiencies by looking at the PowerPoint XML ourselves. Fortunately for us, the python-pptx stores all of the package XMLs as already-parsed lxml.Element objects. These can usually be accessed through the .element or .\_element member of python-pptx objects. For example, to access the slide\_master XML, we simply look at the lxml.Element object at prs.slide\_master.element, where prs is the pptx.Presentation object we got from parsing the PPTX file. Using the xpath function, we can then inspect the children elements of the slide master XMl.