Picture 0  
**APEX Office Print: Custom Word Template**

**{title}**  
**{text}**

The tag: **{title}** will be replaced with Hello World!  
The tag: **{text}** will be replaced with This is an exampl...

Simple angular like expressions  
Other possibilities for == : <, > , <=, >=, == , != .  
Another possibility: {currency == 'EUR' ? 'â‚¬'+price : price+'$'} prints â‚¬50 if true 50$ if false and given price variable is 50.  
**{#title=='Hello World!'}**  
This block will be printed if the previous expression is true.  
**{/title=='Hello World!'}**  
**{#text=='This is an exampl...'}**  
This block will be printed if the previous expression is true.  
**{/text=='This is an exampl...'}**

Image Tags  
If a key has base64 encoded image on it then  **{%imagekey}**  tag can be used to display the image.  
  
Barcode and QR Codes  
QR codes and barcodes can be inserted for a value of a given key by  **{|barcodekey}** tag. The type of QR code or barcode should be specified by 'barcodekey\_type' in json file. Eg:  
If you want to insert qr code for a value of key 'link' then 'link\_type' key should have 'qrcode' as value. The available types are but not limited to:   
- codabar  
- code128  
- code39  
- ean13  
- upc-a  
- upc-e  
- qrcode  
- dotcode  
- See: https://www.apexofficeprint.com/docs/#barcode-qrcode-tags for a more complete list.

Extra information can be given by the following tags:  
**'key\_height' :** contains the height of barcode/qrcode to be generated. default:200 for qrcode, 50 for the rest.  
**'key\_height' :** contains the width of barcode/qrcode to be generated. default: 200   
**'key\_version' :** contains the version of QR code to be used. default: 4  
**'key\_errorcorrectlevel :**contains the level of which the QR code should be recoverable ('L','M','Q' or 'H'). default 'M'  
**'key\_cellsize' :** contains the dot size of a module inside QR code   
Please refer to AOP docs https://www.apexofficeprint.com/docs/#615-barcode-qrcode-tags

htmlcontent  
**{\_htmlcontent}** can be used to write mark-up and AOP will translate the html mark-up into native word markup for you.  
We need to use the tag above in the template file with underscore and define the tag in the data.  
We can use orderlist (ol) followed by its list (li) in the data and AOP converts it to numbered list in word. Similarly 'strong' of HTML is bold in word.   
Refer to AOP docs for more information.

Hyperlink  
HyperLink feature can be added in document to move to another location or simply to a link. **{\*hyperlink}**can be used in template file. The hyperlink text can be given by giving extra data that ends with \_text. e.g: {\*tag\_text}.

Footnote  
**{+footnote}**tag can be used to add footnote on the document. This tag starts with '+' symbol. Refer to the docs of AOP for more information.

Cell Markup Word  
**{data\_string\_with\_cell\_markup$}** tag is used inside table cell. We can define width for the given column by appending **\_width** with the tag value and similarly for background color **\_cell\_background\_color** can be used.  
The data for background color should be hex color and the width can be specified in cm, px, pt, em,'in' and %. When % is in respect to the initial width of the table.

Table of Contents  
We can place **{~tockey}** tag to generate the table of contents at the place where this tag is used.   
The following options are available:   
  
**- Table Title:** '\_title' adds title to the table of contents  
**Show Level:** '\_show\_level' defines the depth to be shown. The default value is 3  
**Tab Leader:** '\_tab\_leader' adds the space between title and page number should be filled. Can be hyphen, underscore, or dot (default).

Breaking into Groups  
With this filter we can break data array into groups based on the group parameter given.Our data remains same. Simple syntax is given below:  
**{#products|break:"CATEGORY"} ... {/products|break:"CATEGORY"}**Here category is a breaking parameter. And on the basis of category we get group of dataTwo parameters can also be passed.First one is for sorting. It has two option asc for ascending order and desc for descending order. Second parameter is positioning the null value group. It has two options first and last. They are positioned as the naming suggests in the beginning or at the end of the document respectively.

Number formatting  
We can format the numbers using angular expressions. The following formats are supported:   
**FML999G999G999G999G990D00**  
**FML999G999G999G999G990**  
**999G999G999G999G990D00**  
**999G999G999G999G990D0000**  
**999G999G999G999G999G999G990**  
**999G999G999G999G990D00MI**  
**S999G999G999G999G990D00**  
**999G999G999G999G990D00PR**  
**FML999G999G999G999G990PR**  
Use the following structure (the following is showing how to format the number given by unit\_price):   
{tag|format:formatMask:decimalSeperators:currency}  
concrete:  
{unit\_price|format:"FML999G999G999G999G990D00":".,":"$"}

Math operations during a loop  
Numeric operations inside a loop are supported by using the | operator. The following operations are supported:  
**|sum**   
**|avg**   
**|min**   
**|max**   
The syntax has been modified to take null values into account.  
**For Example:**  
array|(min|max|sum|avg):"attributeToApplyTo":takeNullValueIntoAccount:valueForNull  
**concretely:**  
{projects |sum:"totalfunding":true:0}  
{projects |max:"totalfunding":false}  
{projects |avg:"totalfunding":false}  
{projects |min:"totalfunding":false}