Customizing the Recommendation Item Template

You can customize the appearance and feel of product recommendations in a Product Recommendations for Email experience by customizing the recommendation item HTML template that's in the email recommendations action template. This HTML template control each product's look and feel and allow you to customize the display of each product's price, title, and all other standard catalog attributes. You can also include conditional display of elements. This formatting is accomplished with the Jinja2 templating engine. All features of this syntax are available to you to create rich recommendations email experiences.

Syntax

On a fundamental level, you can think of each generated recommendation image displayed in each email as a single HTML document that is then rendered to the aforementioned image. It's much like making a small web page and taking a screenshot. With this concept in mind, you can set up recommendations as if you're setting up small web pages using paragraph tags, image tags, CSS, etc.

All standard visual HTML elements are supported for creating templates. Audio and video elements, however, are not. W3Schools is a useful reference for HTML elements.

The template layout can easily reflect the design language of your website. A simple HTML structure may look like this:

```
<div>
  <p>{{title}}</p>
  <img src="">
</div>
```

This example demonstrates a basic HTML structure as well as the use of product attributes within the template.

Product attributes are accessible by the templating engine when generating recommendation product images. For each product, references to the name of the attribute (such as `price`) can be used in the template, leading to that product's actual price as represented in the product catalog being rendered.

You can immediately use the following product catalog attributes within a template, provided that these attributes are populated for all products in your product catalog:

- `id`
- `image_link`
- `item_group_id`
- `link`
- `price`
- `product_type`
- `sale_price`
- `title`

Additionally, you can use several other standard product catalog attributes in templates:

- brand
- description
- additionalImageLink
- ageGroup
- availability
- adult
- ageGroup
- availabilityDate
- color
- condition
- energyEfficiencyClass
- expirationDate
- gender
- isBundle
- loyaltyPoints
- material
- mobileLink
- mpn
- multipack
- pattern
- promotionId
- salePriceEffectiveDateBegin
- salePriceEffectiveDateEnd
- shipping
- shippingHeight
- shippingWidth
- shippingWeight
- shippingLabel
- size
- sizeType
- tax

> Using the attributes in the second list requires backend work by Monetate Development Services. Contact your dedicated Customer Success Manager if you want to use these attributes in Product Recommendations for Email action templates.

The product attribute names themselves are parts of expressions in the template, which is why they appear in the code example inside pairs of curly braces. All expressions must be placed within these braces in the form of `{{`*expression*`}}`, with *expression* serving as a placeholder for the expression content. For example, if you want to reference the product price in a template, then you use the expression `{{price}}`.

Here's an example of that expression along with the product title in an expression within the context of an HTML document:

```
<div>
  <p>{{title}}</p>
  <p>Price: ${{price}}</p>
</div>
```

Expressions aren't limited to the static use of product attributes. You can also use them to format these attributes in interesting ways and even to perform powerful transformative functions.

One of the most common uses for data formatting is to ensure that product prices display with the correct number of decimal places. One way to do this is to use the `format` function:

```
{
  {
    '{:,.2f}' .format(price)
  }
}
```

The `format` function called with a value as its argument—in this case, the item's price—formats that value for display purposes according to the rules defined by the control string. In this example, the control string `{:,.2f}` indicates that no padding should occur in the spacing of the displayed number and that two decimal places always appear. For example, the string `{:10,.2f};` indicates that the spacing of the displayed number can be padded to 10 spaces and that two decimal places always appear. Finally, the string `{:10,.4f}` indicates that the spacing of the displayed number can be padded to 10 spaces and that four decimal places always appear.

The use of the `format` function in the example snippet is the most common. Within the context of a template, it may look like this:

```
<div>
  <p>{{title}}</p>
  <p>Price: ${{'{:,.2f}'.format(price)}}</p>
</div>
```

You can also use math operations in expressions, such as multiplying a number by 2. You accomplish this instruction with the operators +, -, *, and / to represent addition, subtraction, multiplication, and division, respectively. Here's an example:

`{{5*2+1}}`

Product Images

While the customer sees the values of most product catalog attributes (such as the title), product images are stored in the product catalog as links to images rather than as embedded images. For this reason, you *cannot* reference a product image as follows:

`<p></p>`

This code results in the customer seeing the long link to the image instead of the image itself.

Because the templates are handled as HTML, you must reference any image link as an image within the HTML document by using the following code:

```
<img src="">
```

In the above example, an expression passes the value of the product's `imageLink` attribute as the image tag's `src` parameter to ensure that the product's image is rendered as part of the final rendered recommendation image. Note that any image, either stored as a link in the catalog or some external resource, may in theory be referenced from a template, though generally the product's `imageLink` is the most commonly used.

Conditional Statements

Recommendation product templates can use conditional statements for more dynamic Product Recommendations for Email experiences than what otherwise would be possible. One frequent use is to conditionally show "On Sale" badges if a particular item is discounted. Conditional statements themselves contain an "if" statement that includes expressions (for example, to determine if the product's current sale price is lower than its normal price) and extra delimiters to set aside the HTML within the conditional statement. When a conditional statement evaluates to something truthful, the contents of that block of HTML within the conditional statement delimiters are rendered. Otherwise, it doesn't.

The syntax for conditional statements is different from an expression alone because it doesn't use pairs of curly braces but rather `{%` and `%}`. An example, including the conditionally rendered content, is as follows:

```
{% if condition %}
  <p>This text displays conditionally</p>
{% endif %}
```

In this example, `condition` is an expression. Note that `{% endif %}` ends the statement and that the paragraph is that which is conditionally rendered.

Conditional statements usually rely on comparison operators to determine the truthfulness or falseness of an expression. Comparison operators include ==, !=, >, and <, for equals, doesn't equal, greater than, and less than, respectively.

The following conditional statement determines that if the sale price is different from its normal price, then alert the email recipient of a special deal:

```
{% if (salePrice != price) %}
  <p>Special deals may be available!</p>
{% endif %}
```

When building conditional statements, consider including certain content or alternate content in particular situations. Continuing with the previous sale example, imagine that if a product is on sale, then you want the email recipient to see the original price struck out and replaced by the sale price, but if the product isn't on sale, then you want only the regular price to appear. This conditional situation requires the use of `else`. With `else` blocks, either one block of HTML or another is rendered depending on whether the conditional statement is true. Here's an example:

```
{% if (salePrice != price) %}
  <p><s>${{price}}</s></p>
  <p><b>${{salePrice}}</b></p>
{% else %}
  <p>${{price}}</p>
{% endif %}
```

This code takes full advantage of both conditional expressions and HTML. If the sale price doesn't equal the price, the original price appears struck through—note the `<s>` tag—and the current sale price appears in boldface. Otherwise, the original price appears without further adornment.

Style

It's common practice to style HTML documents to meet a particular look and feel, and the HTML documents rendered for recommendations emails may be styled like any other because the documents rendered are effectively following the same rules as those rendered by a browser. Although the specifics of CSS are outside the scope of this documentation, here are certain common use cases and general best practices.

All templates should contain a top-level `<style>` tag and contain overflow prevention on both the X and Y dimensions. This styling ensures that if something is rendered outside the document's viewport, no scrollbars or any other artifacts are baked into the rendered result. A simple example is as follows:

```
<style>
  body {
    overflow-x: hidden;
    overflow-y: hidden;
  }
</style>
```

In addition to top-level styles, individual elements can be styled either by class names, which is recommended, or in simple cases by direct styling of elements. These simple cases are easily illustrated. If you want to render paragraph text in blue, the following code would suffice:

```
<p style="color: blue;">This text is blue!</p>
```

By default, text is black and the background color is white.

Sample Template

The following sample template contains all the pieces discussed in this documentation. You can use it as the basis for your own product recommendations email templates.

```
<style>
  body {
    font-family: serif;
    overflow-x: hidden;
    overflow-y: hidden;
    color: green;
  }
</style>
```

```
<div style="width: auto;">
  <p style="color: black;">{{title}}</p>
  {% if (salePrice != price) %}
  <p><s>${{'{:,.2f}'.format(price)}}</s></p>
  <p><b>${{ '{:,.2f}'.format(salePrice)}}</b></p>
  {% else %}
  <p>${{'{:,.2f}'.format(price)}}</p>
  {% endif %}
  <img style="width: 100%;" src="">
</div>
```