pui.retrieveCustomLayoutTemplate(name)

This function allows you to provide your own custom layout template that can be used to customize a layout widget with your own design.

Parameter:

- **name** - the name of the template to load. Profound UI will expect a file placed under /www/***YOUR-INSTANCE***/htdocs/profoundui/userdata/layouts/***NAME***.html in your HTTP server instance.

In order for this template to be used in the Visual Designer, you will need to load it from a file in your /www/YOUR-INSTANCE/htdocs/profoundui/userdata/custom/widgets directory (or a similar location.)

Example:
Create a file in the IFS named /www/***YOUR-INSTANCE***/htdocs/profoundui/userdata/custom/widgets/loadTemplate.js, and put the following inside:

```
pui.retrieveCustomLayoutTemplate("My Template");
```

Then put the contents of the template under /www/***YOUR-INSTANCE***/htdocs/profoundui/userdata/layouts/My Template.html.

Format Of a Custom Template:

A custom template is an HTML template that describes what your layout will look like. The HTML can contain variables that are evaluated as well as special HTML attributes that control how a given HTML tag is used.

Special attributes:

- **condition** - the condition attribute determines whether an HTML tag is inserted in the page. If the condition evaluates to "false", the HTML element is not inserted into the display, otherwise it is inserted.
- **repeat** - the repeat attribute determines the number of times an HTML tag is repeated. If a repeat attribute evaluates to a positive number, it is repeated, otherwise it will be hidden.
- **container** - when a container attribute evalulates to "true", widgets can be placed inside of this element.
- **stretch** - when a stretch attribute evaluates to "true", this element is stretched to the size of its parent element.

Variables:

Variables in a layout template are enclosed in curly braces, and are either evaluated when the display is rendered. To understand how a variable is used in your template, see "Template Example", below. There are two types of variables that can be provided, they are:

- **JavaScript expression variables** - JavaScript code can be written that returns a value for a variable. A JavaScript expression variable offers the following keywords:
  - **designValue** - JavaScript code that is evaluated when the layout is rendered on the canvas in the Visual Designer.
  - **runtimeValue** - JavaScript code that is evaluated when the layout is rendered at run time.
  - **proxyValue** - JavaScript code that is evaluated when the layout is dragged onto the canvas in the Visual Designer.

- **Property-based variables** - Specifies a property that can be selected in the Visual Designer and will be inserted in place of the variable. Properties can potentially be bound to host variables.
  - **property** - the name of the widget property to evaluate.
  - **help** - the help shown for this property in the Visual Designer.
  - **choices** - the possible choices that can be provided in the Visual Designer.

Template Example:

```
<table style="empty-cells: show; overflow: hidden;" width="100%" height="100%">
  <tr repeat="{ property: 'rows', help: 'Specifies the number of table rows for this layout.' }">
    <td style="border: { designValue: '1', runtimeValue: 0 }px dashed #666666;"
        repeat="{ property: 'columns', help: 'Specifies the number of table columns for this layout.' }">
      <div stretch="true" container="true" style="position: relative; width: 100%; overflow: hidden;">
      </div>
    </td>
  </tr>
</table>
```

Example notes:

- On line 2, the repeat attribute is used to repeat everything in the <tr> element (which includes everything up to the </tr> tag). A variable value is specified for the number of times to repeat the element. The value of the variable is determined by the "rows" widget property.
- On line 3, the standard HTML style attribute is used to control the inline style of the <td> tag. The value of the 'border' style is determined by a JavaScript variable. When run in the Visual Designer, the border width will be 1, but during runtime, it will be 0. This enables dashed lines to display for the table borders in the Visual Designer, while making the borders invisible at run time.
- On line 4, the <td> tag is repeated based on the number of columns selected in the "columns" widget property.
- On line 5, the element is a container, so you can drag widgets into it in the Visual Designer. It is stretched so that it is always the full size of the <td> element that it is inside.