Custom Question Builder

Important Update to Custom Scripting

The CustomScript Action now supports the LUA programming language. Visit ourNEW Lua Scripting Resources!

1. New accounts (created after October 29, 2018) will only have the option to use Lua in scripts.
2. As of October 29, 2018 Custom Scripting Actions will default to Lua as the scripting type in the Custom Scripting Action for accounts created before this date. You will be able to switch to the Legacy Custom Scripting; though we highly encourage using Lua.
3. In the long term, Legacy Custom Scripting Actions will be switched to read-only. The exact date on this is to be determined; we will send notifications well ahead of time.

Custom Questions are not supported with Standard Reporting. They are only supported with Legacy Reporting, which is now depreciated.

Available on these licenses: Full Access

Under **Research Library > Custom Questions**we provide a framework to build your own survey question from scratch. Below we'll cover a fairly basic custom question setup.

Before we get started, you should know that we created custom questions for advanced programmers to choose their own adventure within Alchemer. Part of that adventure is support, as custom questions are beyond the scope of our support team.

In this tutorial we will cover the Net Promoter Score® (NPS®) question that we originally built as a custom question before adding it to the core application.

About Tab

On the **About** tab enter the descriptive details of your custom question. You'll want to make sure you specify, at the very least, a **Question Type Name**, **Status** and **Description**.

Note: If the status of your Custom Question is**In Design**, the question cannot be added to a

> survey for testing purposes. If you need to test your question, set the status to **Active**.

There are also several fields for optional Image URLs. The Icon URL will display on the Build tab of your survey.

Layout Tab

On the **Layout** tab you'll enter the HTML for your question. Below is the HTML for the NPS custom question. Note, you'll need to use Alchemer the following elements within your HTML:

- Alchemer Classes, e.g. sg-question-title, sg-question-number, etc.
- Custom question merge codes, e.g., %elementid%. See Available Merge Codes.
- Alchemer merge codes e.g., [question("number"), id="%elementid%"] [question("title"), id="%elementid%", displaytitle="true"] [question("required"), id="%elementid%"].

```
<div class="sg-question-title">
   <label for="nps-%elementid%">
      <span class="sg-question-number">[question("number"), id="%elementid%"]</span> [question("title"
), id="%elementid%", displaytitle="true"] [question("required"), id="%elementid%"]
   </label>
</div>
<div class="sg-question-options nps-options">
   <div class="sg-control-custom">
      <!-- this is your canvas -->
      <ul id="nps-labels-%elementid%" class="netpromoterscore">
      </ul>
      <ul id="nps-values-%elementid%" class="npsvalues">
      </ul>
   </div>
</div>
```

Custom Question Merge Codes

- **%elementid%** - the question or element id that will be dynamically populated when your question is created.
- **%pageid%** - the page id that will be dynamically populated when your question is created.
- **%surveyid%** - the survey id that will be dynamically populated when your survey and question is created.

Script Tab

The Script tab should include the Javascript to run onload and onsubmit. Below is the Javascipt from the Script tab for the NPS custom question. Note, you'll need to use custom scipt functions, e.g., getCustomProperty and getStorageValue.

```
{
   array_flip: function(trans)
   {
```

```
    var count = 10;
    var key, tmp_ar = [];
    for (key in trans)
    {
        if (trans.hasOwnProperty(key))
        {
            tmp_ar[count] = trans[key];
            count--;
        }
    }
    return tmp_ar;
},
//onload - show the previous response (if any)
onload: function()
{
    //Rearrange order
    var labels = [];
    labels.push(
        '<li><span id="llabel-%elementid%">Extremely Likely</span>10</li>',
        '<li>9</li>', '<li>8</li>', '<li>7</li>', '<li>6</li>', '<li>5</li>',
        '<li>4</li>', '<li>3</li>', '<li>2</li>', '<li>1</li>',
        '<li><span id="rlabel-%elementid%">Extremely Unlikely</span>0</li>');
    var values = [];
    values.push(
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-10" value="10" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-9" value="9" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-8" value="8" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-7" value="7" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-6" value="6" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-5" value="5" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-4" value="4" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-3" value="3" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-2" value="2" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-1" value="1" /></li>',
        '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-0" value="0" /></li>');
    if (this.getCustomProperty("order") == 'Ascending')
    {
        //flip array
        labels = this.array_flip(labels);
        values = this.array_flip(values);
    }
    $("#nps-labels-%elementid%").html(labels.join(''));
    $("#nps-values-%elementid%").html(values.join(''));
    //Ability to set labels
    var llabel = this.getCustomProperty("l_label");
    var rlabel = this.getCustomProperty("r_label");
    if (llabel != null && llabel != '')
    {
        $("#llabel-%elementid%").html(llabel);
    }
    if (rlabel != null && rlabel != '')
    {
        $("#rlabel-%elementid%").html(rlabel);
    }
    if (!window.isEditor)
    {
        var value = this.getStorageValue();
        if (value != '')
        {
            this.getElement("nps-%elementid%-" + value)[0].checked = true;
        }
    }
},
//Store the value from the custom question into this survey element
onsubmit: function()
```

```
    onSubmit: function()
    {
        var value = $("input[name=nps-%elementid%]:checked").val();
        this.setStorageValue(value);
    }
}
```

CSS/JS Tab

The CSS/JS tab should include CSS and any Client Side Javascript you wish to run in survey taking. Below is the CSS from the CSS/JS tab for the NPS custom question.

```
.netpromoterscore {
display: inline-block;
overflow:auto;
margin-bottom:0px;
}
.netpromoterscore li {
width:35px;
float: left;
display:block;
text-align:center;
padding: 3px;
}
.npsvalues li {
width:35px;
float: left;
display:block;
text-align:center;
padding: 3px;
margin-top:0px;
}
.netpromoterscore li span {
display: block;
margin-left: -7px;
margin-top: -30px;
position: absolute;
text-align: center;
width: 50px;
}
.nps-options {
padding-top:35px;
}
```

The Script Tab Vs The CSS/JS Tab

Both the JavaScript you place in the Question Script field as well as in the Client Side Javascript field will be loaded with your question in a survey, in the same <script> tag no less! The primary difference is the built-in functions you have access to from within the Question Script field.

This includes this.getCustomProperty(name) that you're already familiar with, as well as a whole host of others:

this.getOptions()
this.getRenderContext()
this.getQuestionSku()

this.getSectionSku()
this.getSurveyID()
this.getBaseID()
this.getOptionSku(optionName)
this.getCustomProperty(propName)
this.getFormElement(elName)
this.getStorageValue()
this.getMultiStorageValue(optionid)
this.getElement(elName)
this.setStorageValue(elValue)
this.setMultiStorageValue(optionid, elValue)

These functions are actually defined just below your JavaScript (in the same <script> tag), when loading a live survey with your custom question on it.

In addition to the custom functions you have available to you, note the structure of the example Script. This is an object literal, with some automatic behavior built in to the "onload" and "onsubmit" properties. As you may expect, a function you define for the "onload" property will be executed upon page load, and a function you define for the "onsubmit" property will be executed as the page is being submitted. Here's a simplified example you can play around with to understand this better:

```
{
   onload: function() {
      console.log("The survey just loaded!");
   },
   onsubmit: function() {
      alert("The page is being submitted!");
   }
}
```

In contrast, the Client Side Javascript field expects plain JavaScript, and has no special built-in functions or behaviors. The benefit to using this field would be organizing your code, if you happen to have any JavaScript you want to run regardless of custom settings/etc.

Settings Tab

On the Settings tab, specify which sub-header you wish your custom question to display under in the question type dropdown when adding a question.

You will also need to specify whether to provide a UI for answer choices, any default data points and custom settings you wish to provide for question setup.

Custom Settings
Custom settings are added one per line, and follow this convention:

```
name; type; title; options; default
```

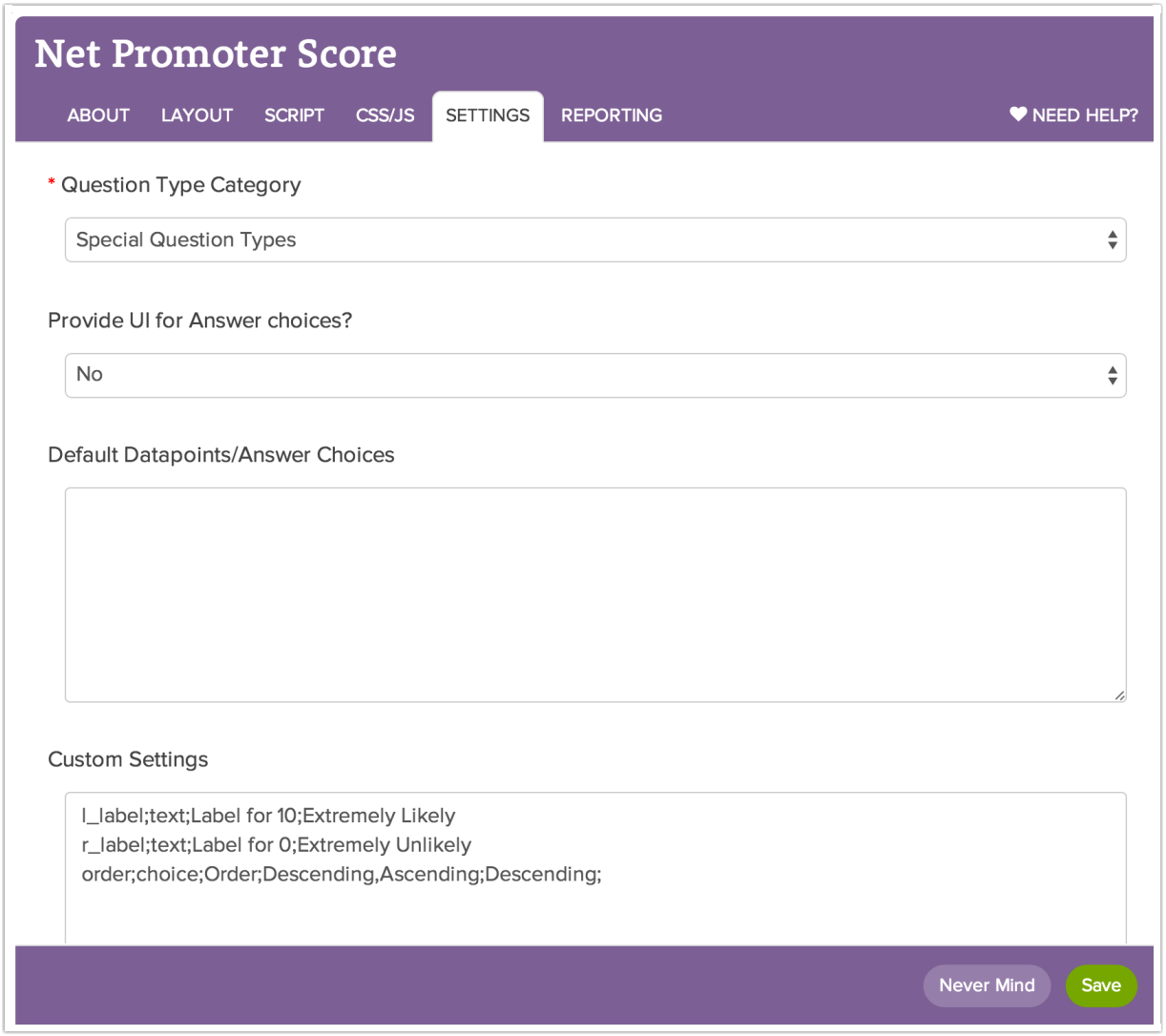
name - This is the name you will use to refer to your custom setting from your "Script," using the this.getCustomProperty(name) function.

type - This determines the type of widget to use for your custom setting. Available options include "text" and "textarea" (open text, in short and long form), "choice" (dropdown menu), and "image" (textbox containing an image URL).

title - This is what will be seen in the Options tab, while editing your question.

options: A comma-separated list of options, only used for the "choice" type.

default: The default value that will be used for your custom setting, if a user does not edit the question and save their own value.



Reporting Tab

Finally, on the Reporting tab you will set up the reporting for your custom question. Indicate whether

the DataType is Single Datapoint or Multiple Datapoints. Under Summary Report Display you can choose to use a Pie Chart, Bar Chart or Table to report. Or, as in the case of the NPS question, select Custom to script the reporting yourself.

```
{
   array_flip: function(trans)
   {
      var count = 10;
      var key, tmp_ar = [];
      for (key in trans)
      {
         if (trans.hasOwnProperty(key))
         {
            tmp_ar[count] = trans[key];
            count--;
         }
      }
      return tmp_ar;
   },
   //onload - show the previous response (if any)
   onload: function()
   {
      //Rearrange order
      var labels = [];
      labels.push(
         '<li><span id="llabel-%elementid%">Extremely Likely</span>10</li>',
         '<li>9</li>', '<li>8</li>', '<li>7</li>', '<li>6</li>', '<li>5</li>',
         '<li>4</li>', '<li>3</li>', '<li>2</li>', '<li>1</li>',
         '<li><span id="rlabel-%elementid%">Extremely Unlikely</span>0</li>');
      var values = [];
      values.push(
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-10" value="10" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-9" value="9" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-8" value="8" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-7" value="7" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-6" value="6" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-5" value="5" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-4" value="4" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-3" value="3" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-2" value="2" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-1" value="1" /></li>',
         '<li><input type="radio" name="nps-%elementid%" id="nps-%elementid%-0" value="0" /></li>');
      if (this.getCustomProperty("order") == 'Ascending')
      {
         //flip array
         labels = this.array_flip(labels);
         values = this.array_flip(values);
      }
      $("#nps-labels-%elementid%").html(labels.join(''));
      $("#nps-values-%elementid%").html(values.join(''));
      //Ability to set labels
      var llabel = this.getCustomProperty("l_label");
      var rlabel = this.getCustomProperty("r_label");
      if (llabel != null && llabel != '')
      {
         $("#llabel-%elementid%").html(llabel);
      }
      if (rlabel != null && rlabel != '')
      {
         $("#rlabel-%elementid%").html(rlabel);
      }
      if (!window.isEditor)
      {
```

```
            var value = this.getStorageValue();
            if (value != '')
            {
               this.getElement("nps-%elementid%-" + value)[0].checked = true;
            }
         }
      },
      //Store the value from the custom question into this survey element
      onsubmit: function()
      {
         var value = $("input[name=nps-%elementid%]:checked").val();
         this.setStorageValue(value);
      }
   }
```

Testing and Debugging

Testing your custom question is simple—add it to a survey! Because both your Question Script and Client Side Javascript are loaded and run on the survey page, debugging is also simple; just call console.log() or alert() from your JavaScript as you see fit.

We also recommend reviewing the data collected by your custom question. A few places to check would be the Individual Responses, in a Summary Report, and in the CSV/Excel Export. This is the primary goal of a survey question, after all—to collect usable data!

Scripting and Other Out-of-the-Box Customizations

We're always happy to help you debug any documented script. That said, we do not have the resources to write scripts on demand.

If you have customization ideas that you haven't figured out how to tackle, we're happy to be a sounding board for Alchemer features and functionality ideas that might meet your customization. Beyond this, you might want to consult with someone on our Programming Services Team; these folks might have the scripting chops to help you to achieve what you are looking for!

- Author: Mel Langworthy
- Share This Article:
- Last updated: 06/28/2023 6:17 pm EDT