

# Custom HTML

- [Introduction](#)
- [Get Started](#)
- [Custom HTML Properties](#)
  - [Edit Custom HTML](#)
  - [Advanced Options](#)
- [Related Tutorials:](#)



## Prevent XSS Attack

When using [Hash Variable](#) that uses URL parameter or user-inputted value in your custom JS scripts, ensure that these hash variable(s) are **escaped**:

Make use of hash variable escape keywords, see [Hash Variable - Escaping the Resultant Hash Variable](#).

Use `?javascript` hash variable escape. Example:

```
#requestParam.id?javascript#
```

## Introduction

**Custom HTML** in Form Builder can be used to achieve advanced form design.

## Get Started

The easiest way to see how the Custom HTML works is to use the existing built-in App Expenses Claims. Here are the steps:

1. Start the **Joget Server** and open the **App Center**.
2. Log in as **admin** and click on the pencil icon on the **Expenses Claim** to open the **App Composer**. (see Figure 1)

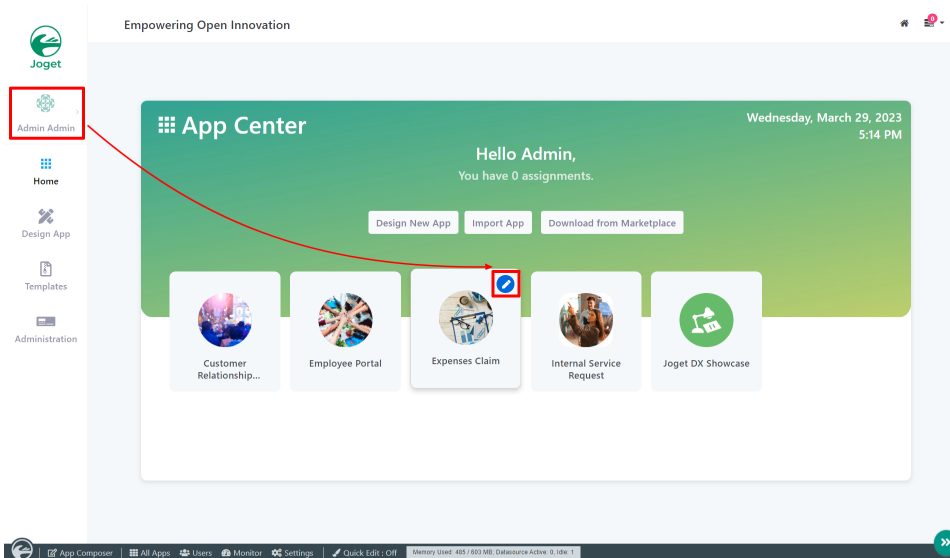


Figure 1

3. Click on **Expense Claim Form** and you will be directed to the **Form Builder**.
4. **Click** on the **Custom HTML** element on the canvas to open up the Configure Custom HTML properties. (see Figure 2).

Take note!

```
<a href="setupCategory" target="_blank">
```

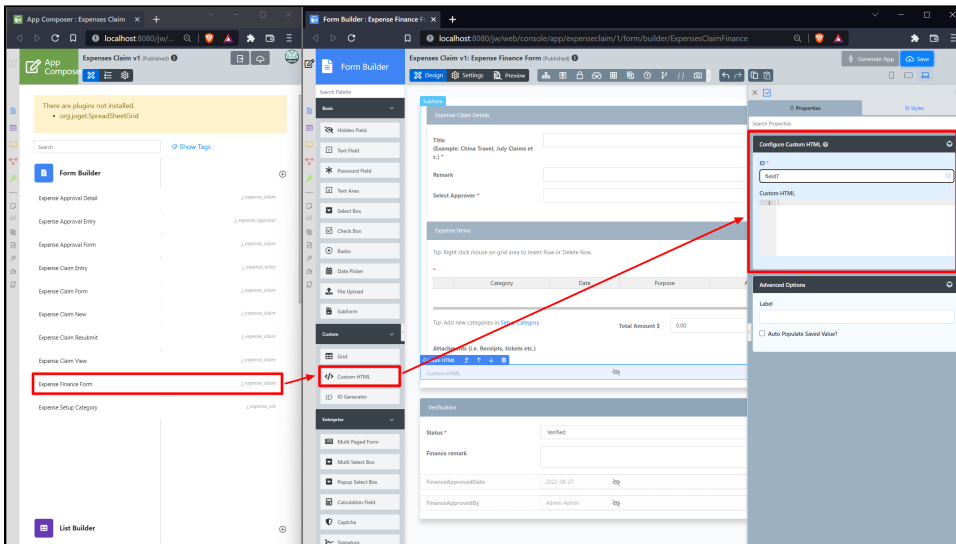


Figure 2

5. This Custom HTML is used to redirect to another page in the App when the user clicks on "Setup Category".
6. To see it working, head back to the **App Composer** and click the **Launch** button in the UI column.
7. Click on **Create a New Expense Claim** button on the **Dashboard**, fill up the necessary details and click **Continue Next Screen**.
8. Here you will find the link. Click it to see it redirects you to the **Setup Category** page.
9. Head back to the **App Composer** and open up the **Expenses Claims Apps** under **UI** column.
10. In **UI Builder** and take a look at the properties of **Setup Category**. Note that the `<a>` href attribute used in Custom HTML element was using the **Menu ID** to specify the page the link goes to. (see Figure 3)

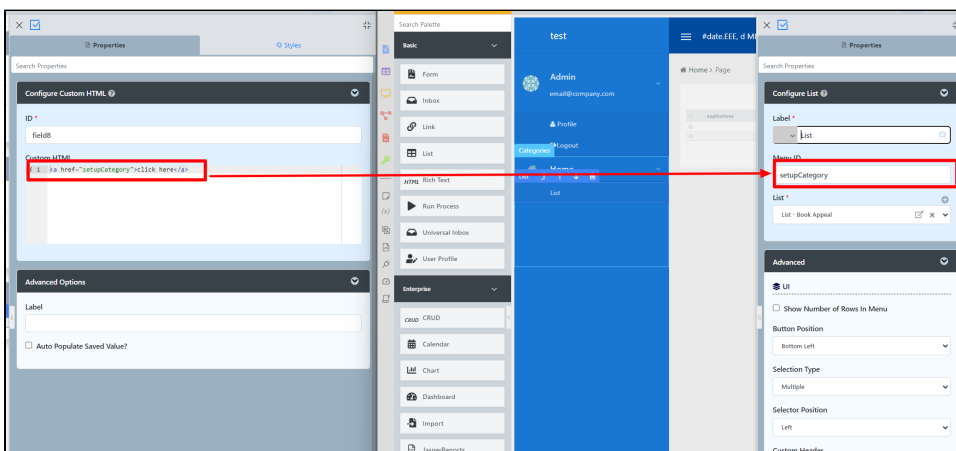


Figure 3

## Custom HTML Properties

### Edit Custom HTML



Configure Custom HTML ?


ID \*

field7

Custom HTML

1

Name	Description
ID	<div><p>Element ID will not be automatically be reflected in the database unless you toggled the <b>Auto populate saved value</b> and use the <code>&lt;input&gt;</code> element in the custom HTML.</p><div><div></div><div><b>The <code>&lt;input&gt;</code> Element</b> Any <code>&lt;input&gt;</code> element in the custom HTML will automatically create a database table column based on the name attribute.  To retrieve the value back, you can enable <b>Auto Populate Saved Value?</b> under Advanced Options below with <code>value</code> attribute available in the code to ensure the value will be stored and fetched in both the form and database..</div></div><p>Please see <a href="#">Form Element</a> for more information about defining the ID and the list of reserved IDs.</p><div><div></div><div><b>Making it Hidden</b> You can name the ID as "hidden" and the content will be hidden away in the runtime/actual UI.</div></div></div>

Custom HTML	<p><b>Custom HTML</b> in Form Builder can be used to achieve advanced form design by putting in any valid -</p> <p>1. <b>HTML</b></p> <div><p>Sample</p><pre>&lt;b&gt;this text is in bold&lt;/b&gt;</pre></div> <div><p>Sample</p><pre>&lt;input type="text" id="fname" name="fname" value=""&gt;</pre></div> <div><p> <b>The &lt;input&gt; Element</b></p><p>Any &lt;input&gt; element in the custom HTML will automatically create a database table column based on the name attribute.</p><p>To retrieve the value back, you can enable <b>Auto Populate Saved Value?</b> under Advanced Options below with <b>value</b> attribute available in the code to ensure the value will be stored and fetched in both the form and database.</p></div> <p>2. <b>JavaScript</b> (jQuery is supported)</p> <p>Don't forget to put in &lt;script type="text/javascript"&gt;&lt;/script&gt; block</p> <div><p>Sample</p><pre>&lt;script type="text/javascript"&gt; alert("hello world"); &lt;/script&gt;</pre></div> <p>3. <b>CSS</b></p> <p>Don't forget to put in &lt;style type="text/css"&gt;&lt;/style&gt; block</p> <div><p>Sample</p><pre>&lt;style type="text/css"&gt; body{   font-size: 100%; } &lt;/style&gt;</pre></div>
-------------	---




## Advanced Options

Advanced Options

Label

☐ Auto Populate Saved Value?

☐ Sanitize Input Value?

Name	Description
Label	Element Label to be displayed to the end-user.
Auto Populate Saved Value?	<p>Toggle to the auto-populate saved value.</p> <div>  <b>The &lt;input&gt; Element</b> <p>Any &lt;input&gt; element in the custom HTML will automatically create a database table column based on the name attribute.</p> <p>To retrieve the value back, you can enable <b>Auto Populate Saved Value?</b> under Advanced Options below with <b>value</b> attribute available in the code to ensure the value will be stored and fetched in both the form and database.</p> </div> <div>  <b>value attribute</b> <p>The Auto populate is based on <b>value</b> attribute. The <b>value</b> attribute must be in the code to ensure the value will be stored and fetched in both the form and database.</p> <p>Eg :</p> <pre>&lt;input type="text" id="myhtml" name="myhtml" value=""&gt;</pre> </div> <div>  <p>Does not support the following input types: file, button, submit, reset &amp; image</p> </div>
Sanitize Input Value?	Checking the box will sanitize the input value before storing input data in the database. Please see <a href="#">Form Input Sanitization</a>

## Related Tutorials:

- [Form Input Sanitization](#)