# Custom Adaptive Form Components


SmartDoc Technologies Courseware: www.aemforms.training

Last Updated: 3-27-2023

# Custom Adaptive Form Components



# Before you Begin

AEM Forms includes a library of *out-of-the-box* adaptive form core components. We can also create our own custom adaptive form components and use them in our adaptive forms. This course will introduce you to adaptive form components and show you how to create your own custom adaptive form components.

## Course Overview

**Prerequisites:** Introduction to Adaptive Forms, Create Adaptive Forms, or similar experience.

**Approximate Instructor-Led Classroom Duration:** 1 hours

## www.aemforms.training

The support site (**www.aemforms.training**) is designed to support our students during and after a training session. Here is what you will find on the support site.

- The **Known Issues** section documents bugs and issues with various versions of AEM Forms.
- The **Sandbox** section lists AEM Forms Servers you can use for the hands-on exercises.
- The **Forum** section enables you to post, review, and answer questions about AEM Forms.

## Limited Courseware License



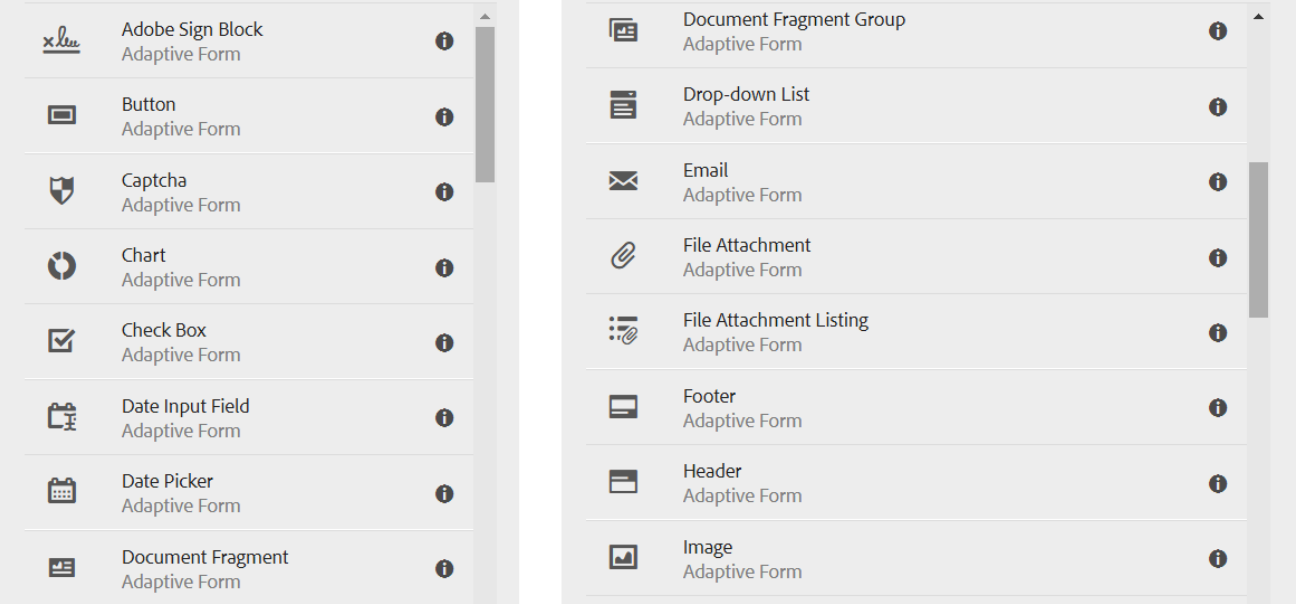
 

# Adobe's Adaptive Form Components

In this module, you will work with and learn about the AEM Forms adaptive form components.

## The out-of-the-box Components

AEM Forms includes a library of *out-of-the-box* adaptive form core components. You can also create your own custom adaptive form components and use them in your adaptive forms. This course will introduce you to adaptive form components and show you how to create your own custom adaptive form components.




Some of the *out-of-the box* adaptive form components.

## Our Custom Components

We will use *out-of-the-box* components to create our own custom components

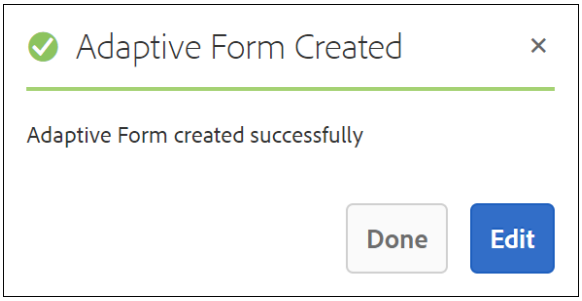
- the Text Box component will become our DynamicCaption component
- the Drop-down component will become our US States component
- the Image component will become our Image Panel component
- the Button component will become our Color Button component

## Exercises

### Create a form

Follow these steps to create a form.

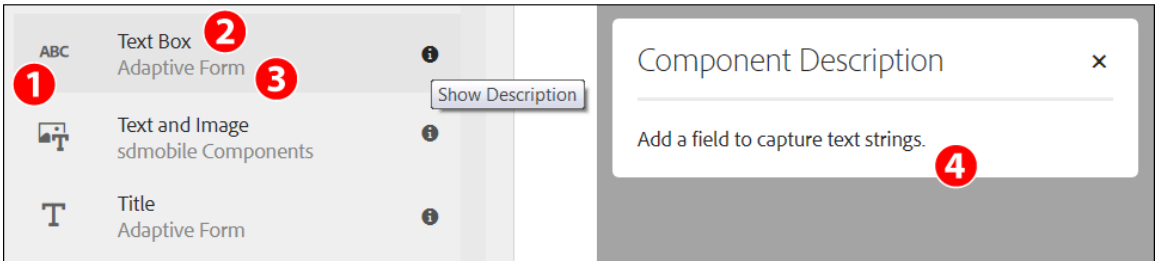
1. Open **Forms – Forms & Documents** and navigate to your working folder.
2. Select **Create – Adaptive Form**.
3. Select the **Blank** template.
4. Click **Next**.
5. Enter **<yourname>SimpleForm** for the Title. The Name will be created automatically.
6. Click **Create**.



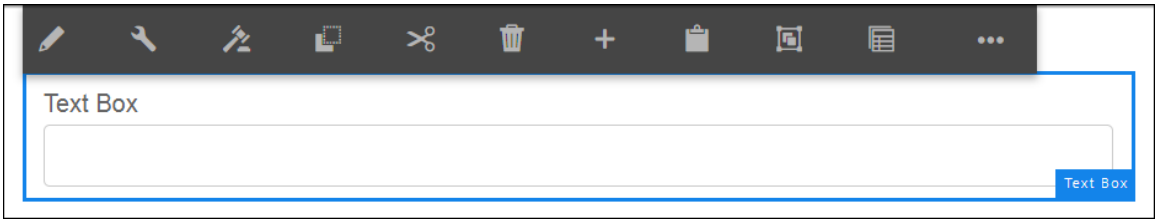

7. Click **Edit**.

## Use the Adaptive Form components

8. Make sure you are in **Edit** mode.
9. Select **Components** on the left.
10. Select **Adaptive Form** to filter your components list. You should just see the *out-of-the-box* AEM Forms adaptive form components.
11. Scroll down to the **Text Box**.
12. You will see the cq:icon (#1).
13. You will also see the jcr:title (#2) and the componentGroup (#3).
14. Click **Show Description** (the icon) and you will see the jcr:description (#4).




15. Drag and drop a **Text Box** component to your form.
16. Select the **Text Box** and you will see a toolbar in Author mode.

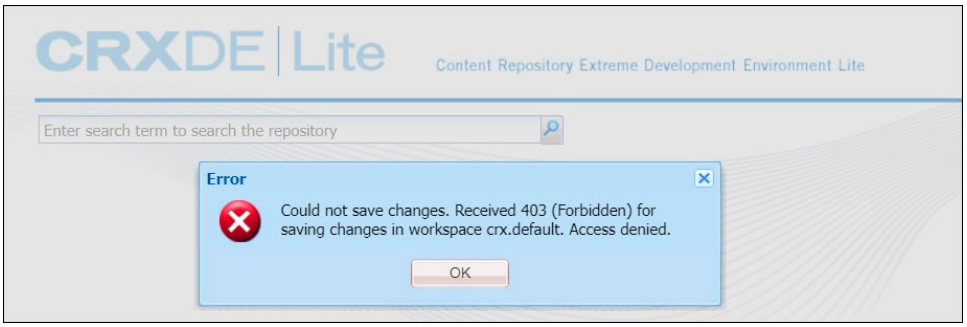



17. Click **Configure** *(the wrench icon)* and you will see Property panels on the left. These property panels are an indication that this component has a cq:dialog child node.

# Create an Application

We will create an application in this section with CRXDE Lite.

***Note:*** *If you are working on advanced topics and need CRXDE | Lite access to the Sandbox, email us (info@smartdoctech.com) and we will extend your privileges. Without these extended privileges you won't be able to save your changes in CRX.*




## The JCR Folders

If you are new to development in CRXDE | Lite, the following definitions are important to know. In particular, we will be copying nodes from Adobe's libs folder and developing them in our apps folder in this course.

| Folder | Description |
|---|---|
| /apps | This is where we create our applications. This folder can contain components, overlays, client libraries, bundles, i18n translations, and static templates. |
| /conf | This stores the configurations, dynamic templates, and policies for application. |
| /content | The content created for your website. |
| /etc | This stores resources related to utilities and tools. |
| /home | This stores the AEM Users and Groups. |
| /libs | This is where Adobe stores the AEM software. It contains the libraries and software for AEM and represent the *out-of-the-box* AEM features. When you apply service packs and cumulative fix packs, you will see many updates to this folder. |
| /oak:index | This stores the Jackrabbit Oak index definitions. Each node specifies the details of one index. The standard indexes are visible, and you can use them to create custom indexes. |
| /system | This folder is only used by Apache Oak. |
| /tmp | This folder serves as a temporary working area. |
| /var | This stores system files including statistics and audit logs. You can find the Java servlets in the /var/classes folder. |

# Exercises

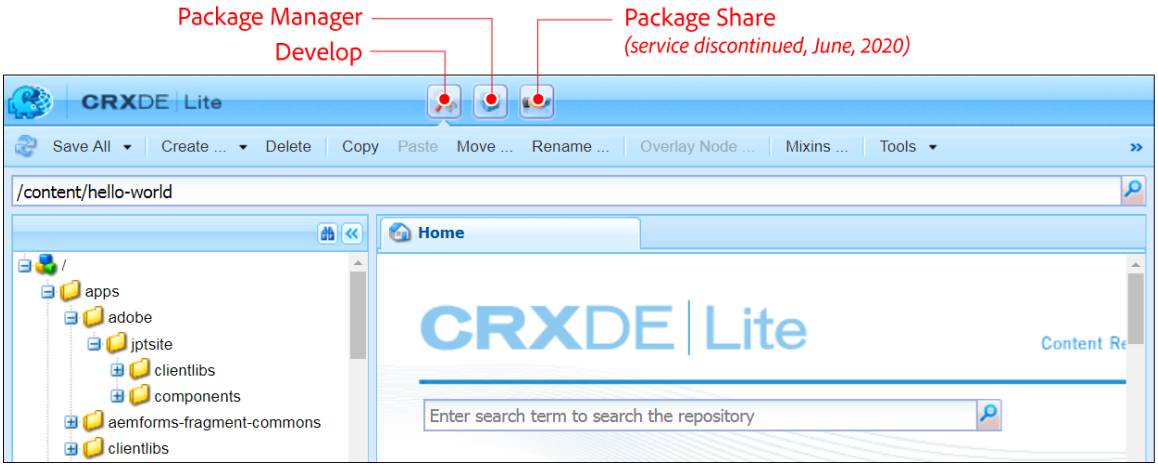
## Create an application folder

***Note:*** *You may already have an application folder on the SmartDoc Sandbox. If you do, you can use this existing application folder.*

1. Navigate to **Adobe Experience Manager – Tools – CRXDE Lite**.

**http://<servername>:4502/crx/de**

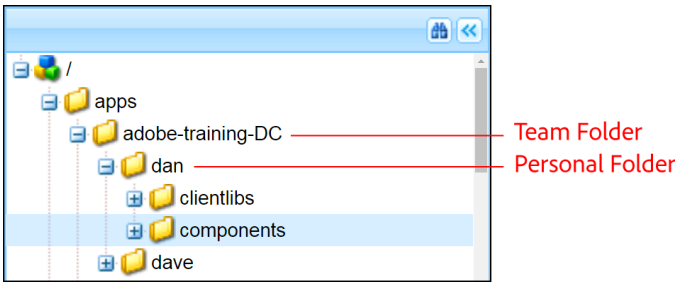
2. Click the **Develop** icon.




3. Navigate to **apps**. Your instructor may create a team folder of type **nt:Folder** for all students to work in. If you don't have this folder, you can create a team folder by right-clicking on apps and selecting **Create – Create Folder**. Provide the name of your company or organization for this team folder.

4. Right-click on **apps/<teamfolder>** and select **Create – Create Folder…**

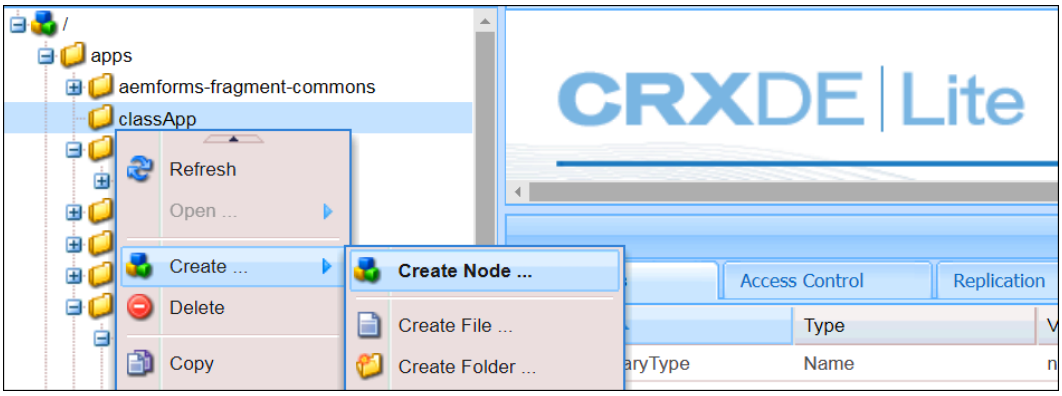
5. Enter **<yourname>** for this personal folder. For instance, in this example, *dan* is the personal folder name.




## Create a Client Library

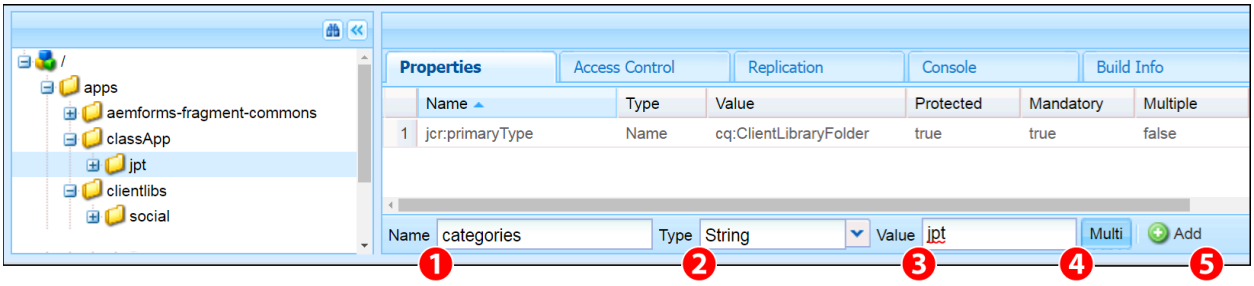
***Note:*** *You may already have a Client Library on the SmartDoc Sandbox. If you do, you can use it. Just review this section to make sure you have everything necessary for your Client Library.*

6. Right-click on **apps/<teamfolder>/<personalfolder>** and select **Create – Node…**




7. Enter **clientlibs** for the Name.
8. Select **cq:ClientLibraryFolder** for the Type.
9. Click **OK**.
10. Click **Save All**.
11. Select the **clientlibs** node and open the Properties panel.
12. Add these values.

– Name: **categories**
– Type: **String**
– Value: **<yourname>**
– Multi: **selected**




***Note:*** *If you are new to CRXDE | Lite, review these bullet points because they will help you when it comes time to add* properties to your node. You add properties with the toolbar at the bottom of the Properties panel.

– #1: This is where you will enter the Name of your new property.
– #2: This is where you will enter the datatype of your new property.
– #3: This is where you will enter the Value of your new property.
– #4: If your new property needs to be an array of your datatype, select the **Multi** button.
– #5: Once your values are all set, click the **Add** button.

13. Click **Add**.
14. Click **OK** in the **Edit categories** window.

15. Click **Save All**.
16. Right-click on the **clientlibs** node and select **Create – Create Folder…**
17. Enter **css** for the Name and click **OK**.
18. Click **Save All**.
19. Right-click on the **clientlibs** node and select **Create – Create File…**
20. Enter **css.txt** as the filename.
21. Click **OK**.
22. Click **Save All**.
23. Double-click the **css.txt** file and enter this code into the text editor.

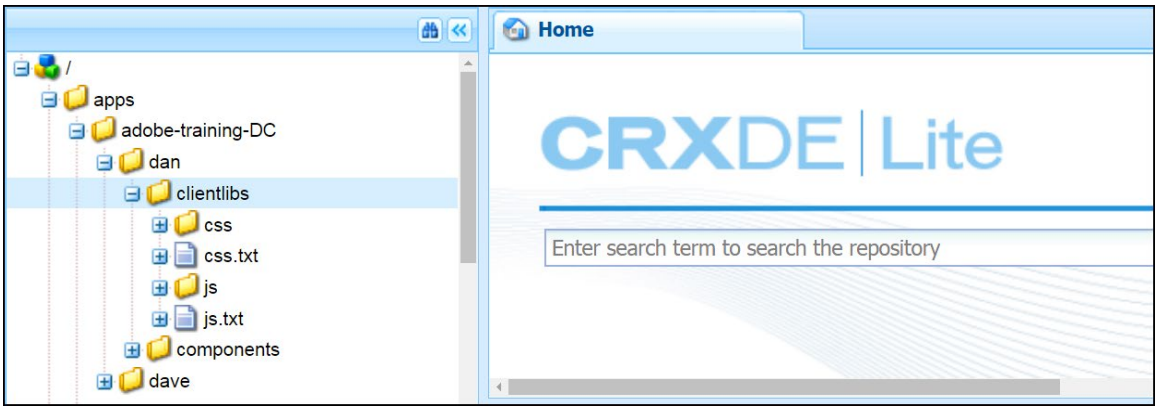
**#base=css**

24. Click **Save All**.
25. Right-click on the clientlibs node and select **Create – Create Folder…**
26. Enter **js** for the Name and click **OK**.
27. Click **Save All**.
28. Right-click on the clientlibs node and select **Create – Create File…**
29. Enter **js.txt** as the filename.
30. Click **OK**.
31. Click **Save All**.
32. Double-click the **js.txt** file and enter this code into the text editor.

**#base=js**

33. Click **Save All**.

Your structure should look like this.



## Create a Components folder

34. Right-click on **apps/<teamfolder>/<personalfolder>** and select **Create – Folder…**
35. Enter **components** for the Name.
36. Click **OK**.
37. Click **Save All**.

You now have an application folder for your custom components.

# Custom Adaptive Form Components

Although every adaptive form component can be unique, they do share many common nodetypes and properties. This section will introduce you to some of these nodetypes and properties and describe the general anatomy of an adaptive form component.

## Adaptive Form Component Attributes

The following is a list of attributes that describe a good adaptive form component.

- An Adaptive form component should be modular and reusable.
- An Adaptive form component should be self-contained.
- An Adaptive form component should have editable properties that a form Author can set.
- An Adaptive form component should have dialogs that are developed with Granite UI components, so they are consistent with AEM's Touch User Interface.

## Nodetypes and properties

The following is a list of nodetypes used in adaptive form components.

- **cq:Component:** The primary nodetype of a component. This is the root node of a component.
- **jcr:title:** The title of the component. The value of this property is displayed to the Form Author in AEM.
- **jcr:description:** The value of this property is displayed to the Form Author in AEM when they click the icon associated with your component.
- **cq:icon:** The value of this property should be a standard icon in the Coral UI library. It will appear to the Form Author in AEM.

## Edit and Config Nodes

The following is a list of nodetypes used in create Edit and Design dialogs in adaptive form components.

- **cq:dialog (nt:unstructured):** Creates a dialog for Edit mode.
- **cq:design_dialog (nt:unstructured):** Creates a dialog for Design mode.

## CoralUI Icons

This table shows a few of the icons in the Coral UI library.

| Icon | Name | HTML attribute |
|---|---|---|
|  | textAdd | icon="textAdd" |
|  | textSize | icon="textSize" |
|  | usa | icon="usa" |
|  | imageAdd | icon="imageAdd" |

A complete list of Coral UI icons can be found at this URL.

**https://helpx.adobe.com/experience-manager/6-5/sites/developing/using/reference-materials/coral-ui/coralui3/Coral.Icon.html#availableIcons**

## JSP

The components get most of their functionality from a JSP page called widget.jsp. We will update these JSP pages to modify the look or functionality of our component. In this example, we are adding a taglib to the JSP that references our client library.

```
  TextBox Component
--%>
<%@include file="/libs/fd/af/components/guidesglobal.jsp"%>
    <c:if test="${isPreviewMode && guideField.allowRichText}">
        <ui:includeClientLib categories="form.richtexteditor"/>
    </c:if>
    <%-- todo: In case of repeatable panels, please change this logic at view layer --%>

    <%@taglib prefix="ui" uri="http://www.adobe.com/taglibs/granite/ui/1.0" %>
    <ui:includeClientLib categories="<your client Library category>"/>
```

## Java Class Files

You can use Java class files for your functionality. AEM has a servlet engine and it will compile our Java class files into Java bytecode.

```
Welcome    package libs.fd.af.components.guideimage Untitled-2 ●
1  package libs.fd.af.components.guideimage;
2
3  import java.io.IOException;
4  import java.io.InputStream;
5
6  import javax.jcr.Property;
7  import javax.jcr.RepositoryException;
8  import javax.servlet.http.HttpServletResponse;
```

# Exercises

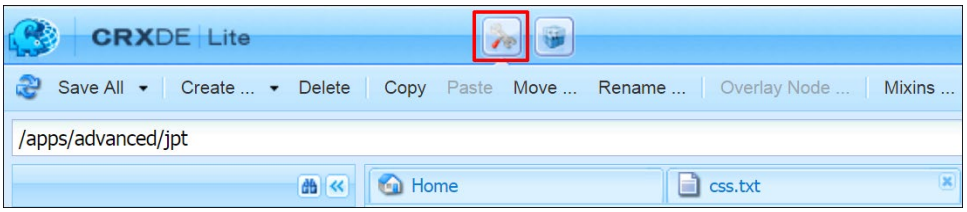
## Your own Custom Text Box

This component will be very similar to the standard out-of-the box Adobe guidetextbox.

1. Open **CRXDE | Lite** if it is not already opened.

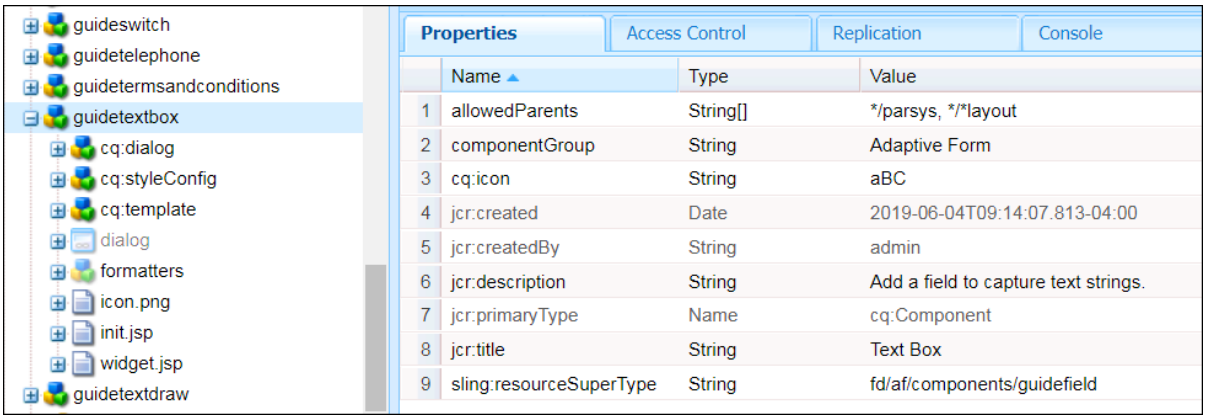
**http://<servername>:4502/crx/de**

2. Click the **Develop** icon.




3. Navigate to **/libs/fd/af/components**.

4. Select the **guidetextbox** node. Notice the child nodes and properties.




5. Right-click the **guidetextbox** node and select **Copy**.

6. Go back to **apps/<teamfolder>/<personalfolder>/components** and select **Paste**.

7. Click **Save All**.

8. Right-click the **guidetextbox** node and select **Rename**.

9. Enter **<yourname>TextBox** as the name.

10. Click **Save All**.

11. Make sure **<yourname>TextBox** is selected so you can see its properties on the right.

12. Enter **<yourname> Custom Components** as the componentGroup.

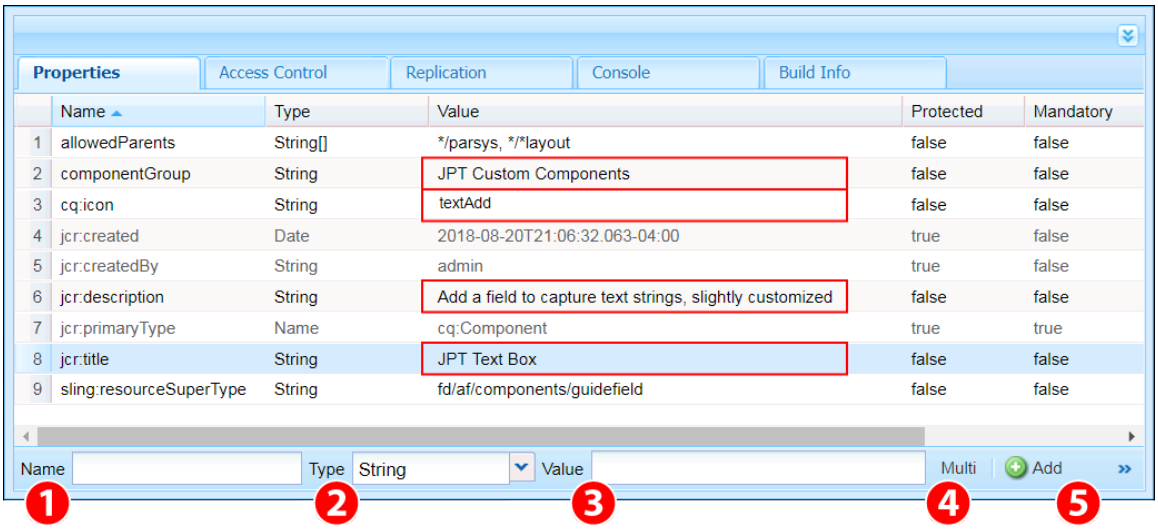
13. Enter **textAdd** as the *cq:icon*.

14. Enter **Add a field to capture text strings, slightly customized** as the *jcr:description*.

15. Enter **<yourname> Text Box** as the jcr:title.

16. Click **Save All**.

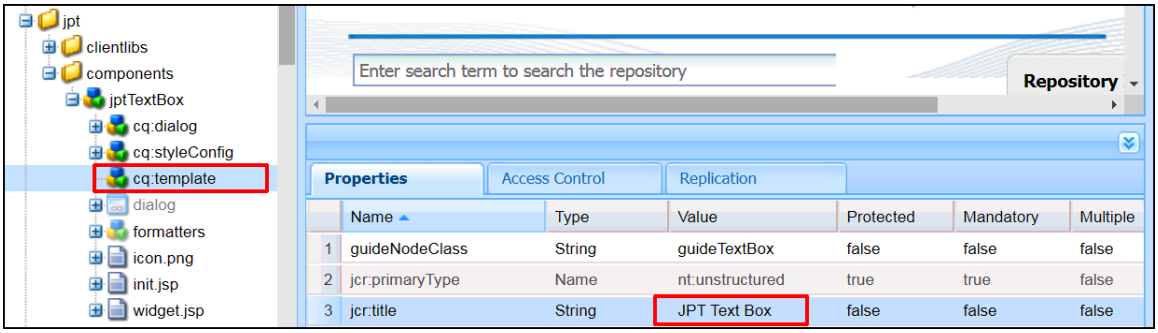
Your properties should now look like this.




***Note:*** *If you are new to CRXDE | Lite, review these bullet points because they will help you when it comes time to add properties to your node. You add properties with the toolbar at the bottom of the Properties panel.*

- #1: This is where you will enter the Name of your new property.
- #2: This is where you will enter the datatype of your new property.
- #3: This is where you will enter the Value of your new property.
- #4: If your new property needs to be an array of your datatype, select the **Multi** button.
- #5: Once your values are all set, click the **Add** button.

17. Expand the **<yourname>TextBox** node and select the **cq:template** child node.




18. Enter **<yourname> Text Box** as the *jcr:title*.

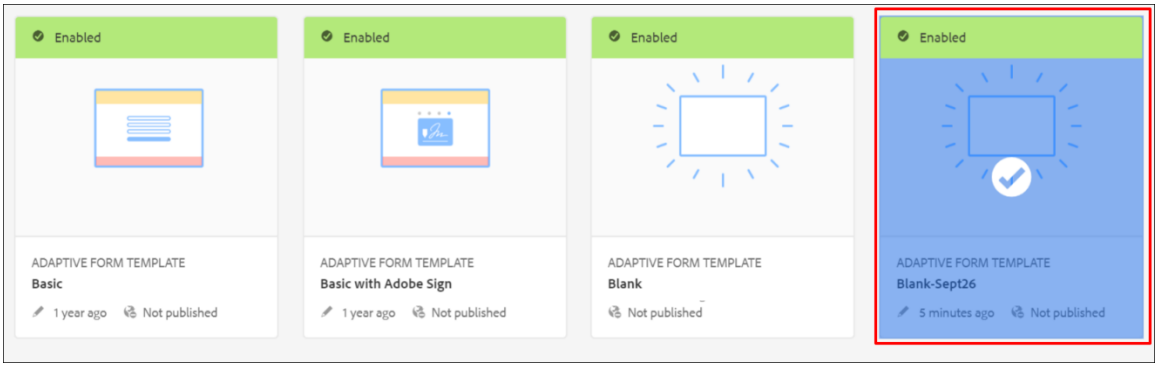
19. Click **Save All**.

## Instructor Task

If you are taking this course as Instructor-led training, your Instructor will complete these steps to create a unique version of the adaptive form template for your class.

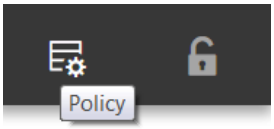
20. Select **Tools – Templates – Adaptive Form Templates – Reference**.
21. Select the **Blank** template and click **Copy**.
22. Click **Paste**.
23. Select the new template and click **Properties**.




24. Enter **Class Template** as the Template Title.
25. Click **Save & Close**.
26. Select **Class Template** and click **Edit** in the Action bar.

    ***Note:*** *In order to see your components in Edit mode, your component group must be added to the new template's Content Policy. Follow these steps to add your component group.*

27. Make sure you are in **Structure** mode.
28. Select the **Adaptive form container** and click **Policy**.



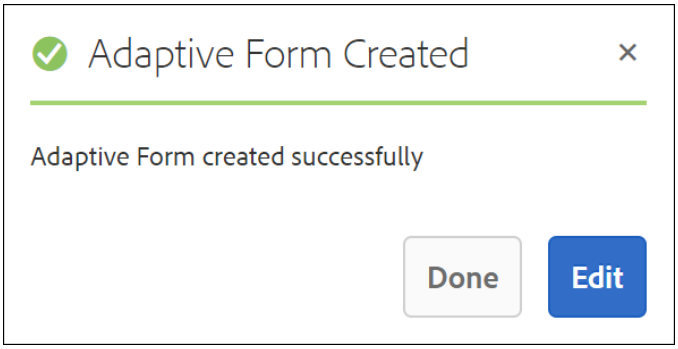

29. Select your component group.
30. Click **Done** to add it to the Policy.

    ***Note:*** *The goal is to have all the new Component Groups added to the Content Policy.*

31. Make sure to enable the template when you finish.

## Add the component to your form

1. Open **Forms – Forms & Documents** and navigate to your working folder.
2. Select **Create – Adaptive Form**.
3. Select the **Class Template** template.
4. Click **Next**.
5. Enter **<yourname> Form for Custom Components** for the Title. The Name will be created automatically.
6. Enter **This form uses the template that includes custom component groups** for the Description.
7. Click **Create**.

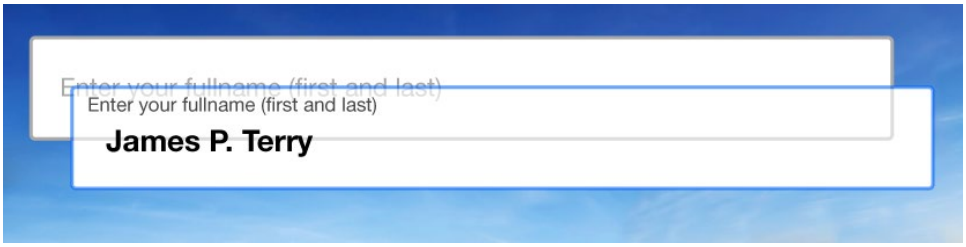



8. Click **Edit**.
9. Select **Components** on the left.
10. Select **<yourname> Custom Components** to filter your components list. You should just see the components in your custom component group when this filter is applied.
11. Scroll down to the **<yourname> Text Box**.
12. You will see the cq:icon (#1).
13. You will also see the jcr:title (#2) and the componentGroup (#3).
14. Click **Show Description** (the icon) and you will see your updated jcr:description (#4).

## DynamicCaption Text Box

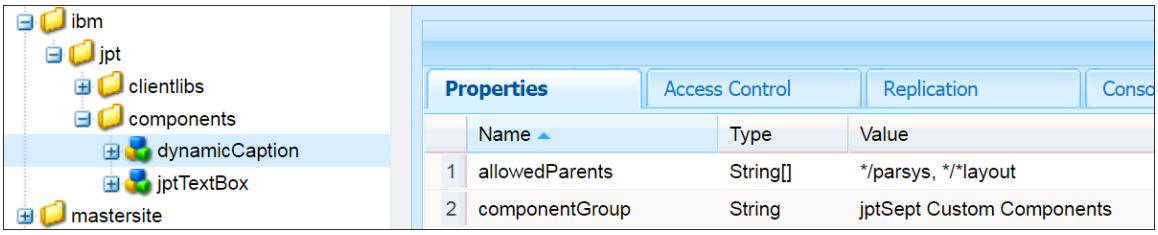
This component will dynamically resize and reposition the caption of a Text Box when the user enters data.




1. Open **CRXDE | Lite** if it is not already opened.

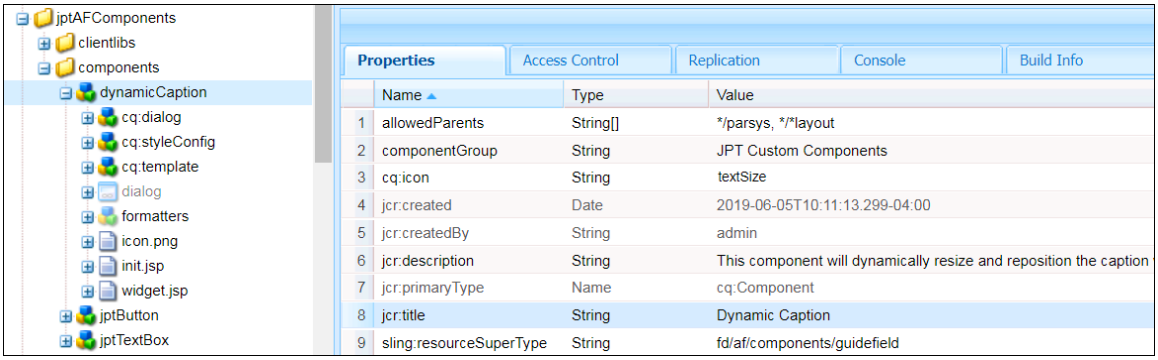
**http://<servername>:4502/crx/de**

2. Navigate to **/libs/fd/af/components**.
3. Right-click the **guidetextbox** node and select **Copy**.
4. Go back to **apps/<teamfolder>/<personalfolder>/components** and select **Paste**.
5. Click **Save All**.
6. Right-click the **guidetextbox** node and select **Rename**.
7. Enter **dynamicCaption** as the name.



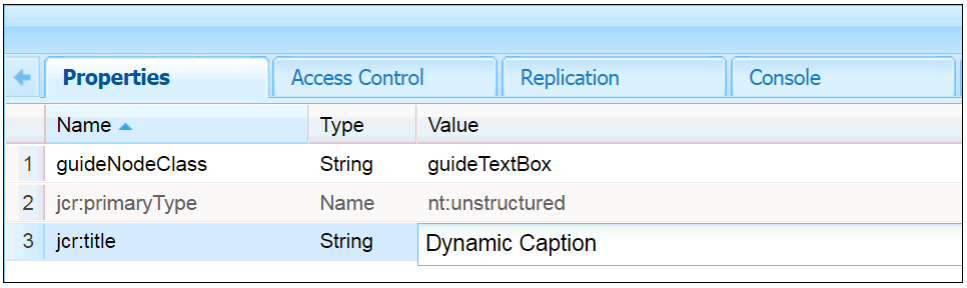

8. Click **Save All**.
9. Make sure **dynamicCaption** is selected so you can see its properties on the right.
10. Enter **<yourname> Custom Components** as the componentGroup. Make sure to use the same name each time for your componentGroup.
11. Enter **textSize** as the cq:icon.
12. Enter **This component will dynamically resize and reposition the caption when the user enters data** as the jcr:description.
13. Enter **Dynamic Caption** as the jcr:title.
14. Click **Save All**.

Your application should now look like this.




15. Expand the **dynamicCaption** node and select the **cq:template** child node.

16. Enter **Dynamic Caption** as the jcr:title.




17. Click **Save All**.

## Update the JSP

18. Expand the **dynamicCaption** node if it is not already expanded.

19. Double-click **widget.jsp** to open the JSP code.

20. Add this taglib and tag to widget.jsp directly below the **<%-- todo:** comment and before the ** **. This taglib and tag can be found in the **taglib.txt** file in your Student Files.

```
<%@taglib prefix="ui" uri="http://www.adobe.com/taglibs/granite/ui/1.0" %>
<ui:includeClientLib categories="<your client Library category>"/>
```

*Note: This is your client library category, not your componentGroup.*

21. Update the categories value with the name of your client library category.

22. Click **Save All**.

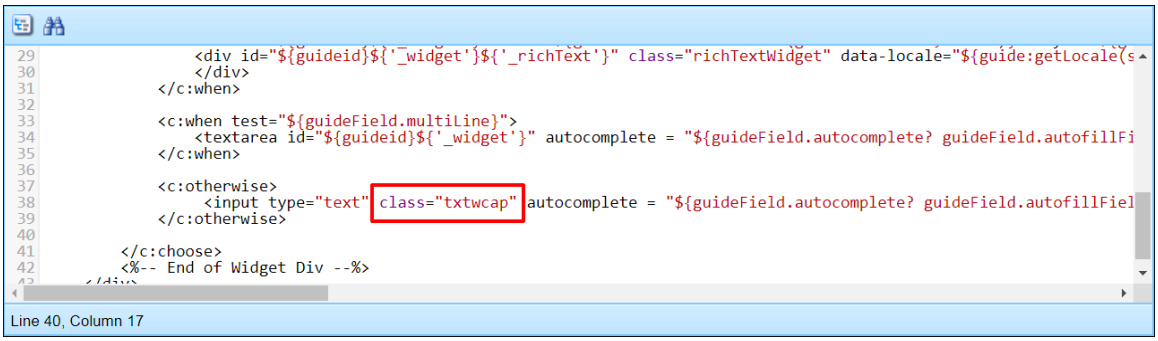
Your code should look like this.


This JSP tag will ensure that the JSP file uses the client library you created earlier. This is the standard CRXDE | Lite method of referencing a client library from a JSP file. You can add JavaScript and CSS selectors to your client library and this JSP page will be able to use them.

23. Scroll to the bottom of the file and add the **class="txtwcap"** attribute to the **input** element of **c:otherwise**.

Your code should look like this.




24. Click **Save All**.

## Add the JavaScript to the Client Library

25. Expand your client library folder.

**apps/<teamfolder>/<personalfolder>/clientlibs**

26. Right-click on the **js** folder and select **Create – Create File**.

27. Enter **dynamiccaption.js** for the *Name* and click **OK**.

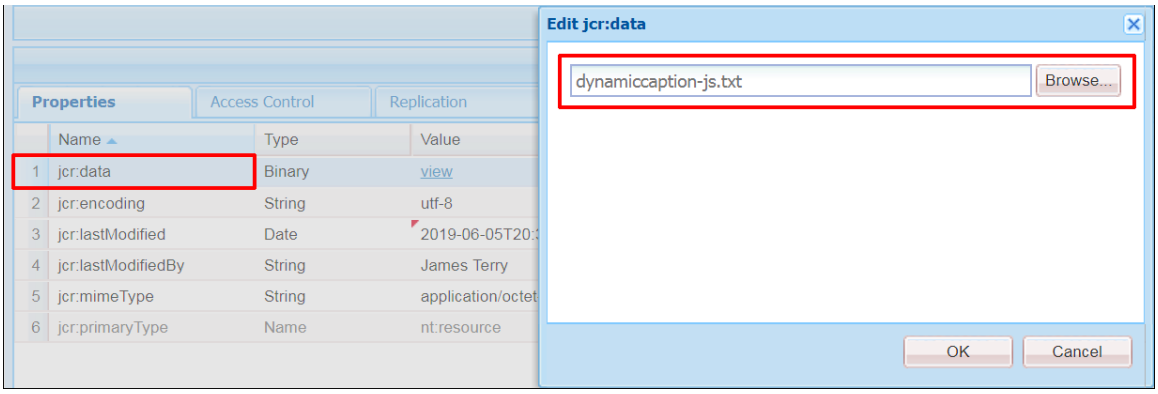
28. Click **Save All**.

29. Expand **dynamiccaption.js** so you can select its **jcr:content** node.

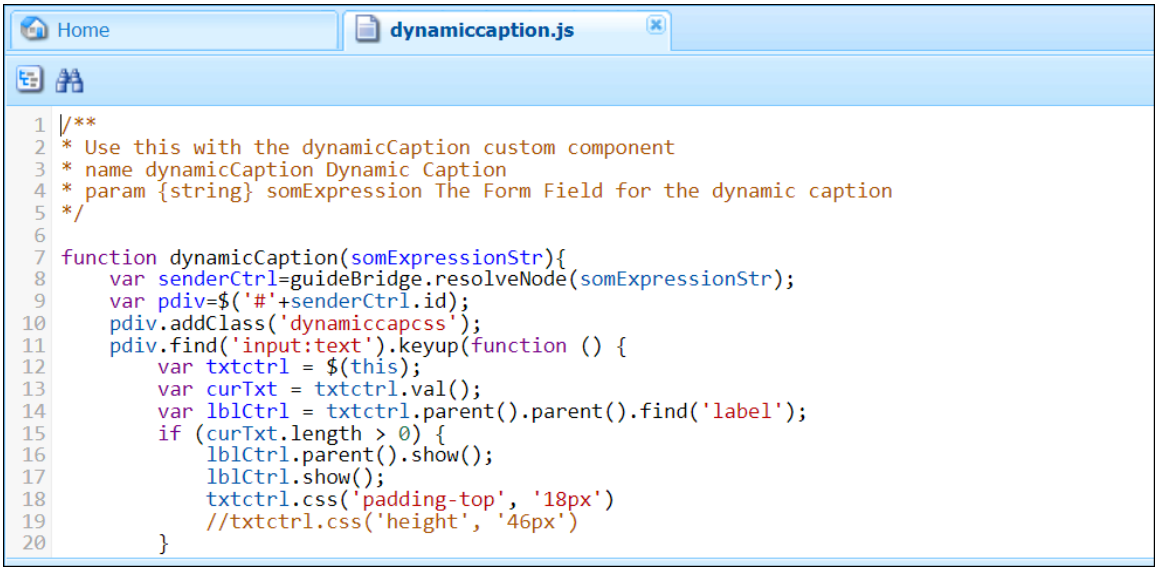
30. Double-click the **jcr:data** property *(see illustration)*.

31. Click **Browse** and navigate to your Student Files.

32. Select the **dynamiccaption-js.txt** file in and click **Open** and you will see the file reference in your dialog box *(see illustration)*.



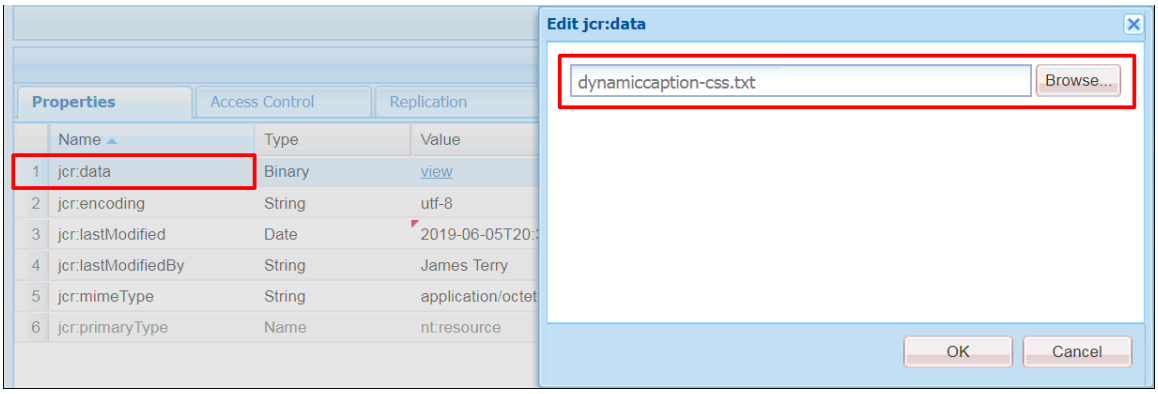

33. Click **OK** and the JavaScript will load into the node.




34. Click **Save All**.

## Add the CSS to the Client Library

35. Right-click on the **css** folder and select **Create – Create File**.
36. Enter **dynamiccaption.css** for the *Name* and click **OK**.
37. Click **Save All**.
38. Expand **dynamiccaption.css** so you can select its **jcr:content** node.
39. Double-click the **jcr:data** property *(see illustration)*.
40. Click **Browse** and navigate to the **dynamiccaption-css.txt** file in your Student Files.
41. Select the file and click **Open** and you will see the file reference in your dialog box *(see illustration)*.
42. Click **OK** and the CSS will load into the node.




43. Click **Save All**.

44. Update your **css.txt** file with a reference to your new file.

**#base=css**
**dynamiccaption.css**

45. Update your **js.txt** file with a reference to your new file.

**#base=js**
**dynamiccaption.js**

## Another way to Reference Client Libraries

Although the previous method is considered best practice, there is another way to reference client library files. Once you have your client libraries established, you can user this approach to reference them in either JSP or HTML pages. You can reference a CSS file with an href link.

**<link href="/apps/<teamfolder>/<personalfolder>/clientlibs/css/dynamiccaption.css" rel="stylesheet"/>**

You can reference a JavaScript file with the script tag.

**<script src="/apps/<teamfolder>/<personalfolder>/clientlibs/js/dynamiccaption.js"></script>**

## Add the component to your form

46. Go back to **AEM Forms**.

47. Open the **<yourname> Form for Custom Components** form and make sure it is in **Edit** mode.

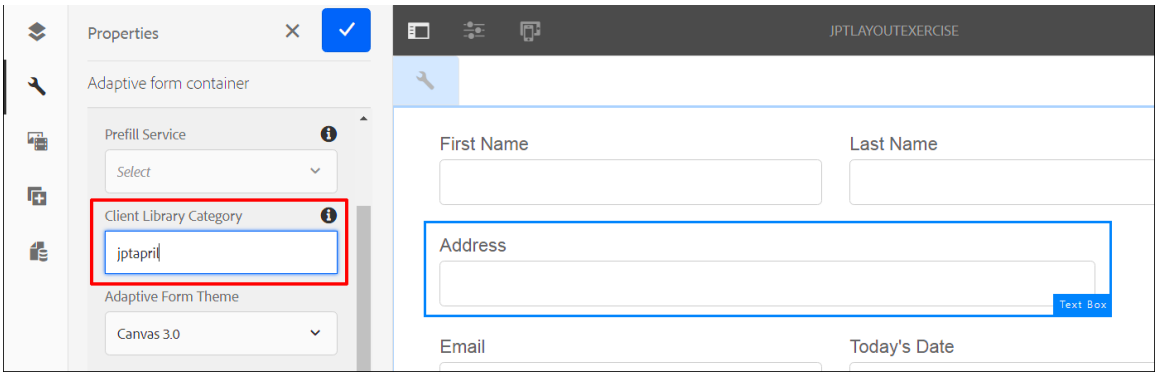
48. Click **Refresh** or **Reload this page** in your browser.

49. In the Side Panel click **Content**.

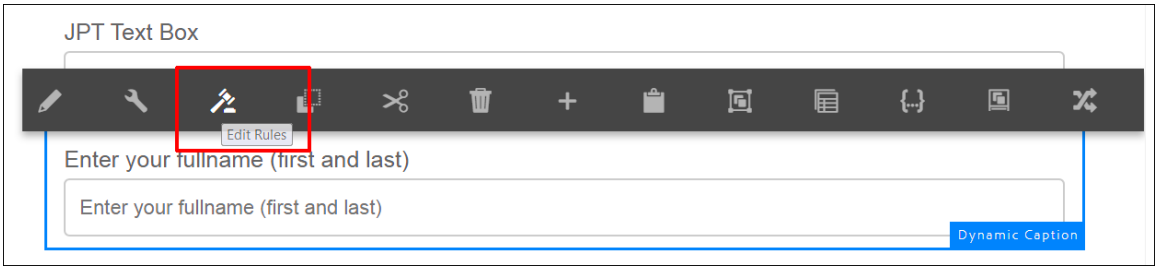
50. Select the **Form Container** at the top of your hierarchy.

51. Click **Configure** (*the wrench icon*).

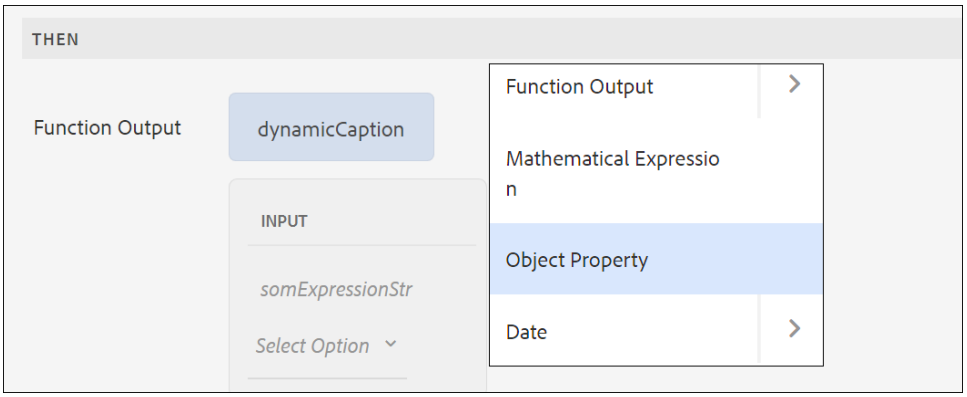
52. In the *Client Library Category* enter **<yourname>** *(the name of your client library category)*.




53. Click **Done**.
54. Select **Components** on the left.
55. Select **<yourname> Custom Components** to filter your components list. You should just see the components in your custom component group when this filter is applied.
56. **Drag and Drop** the **Dynamic Caption** component to your form.
57. Select the **Dynamic Caption** component and click **Configure** *(the wrench icon)*.
58. Enter **dynamiccaption** for the *Name*.
59. Enter **Enter your fullname (first and last)** for the *Title*.
60. Enter **Enter your fullname (first and last)** for the *Placeholder Text*.
61. Click **Done**.
62. Select the **dynamiccaption** field and click **Edit Rules**.



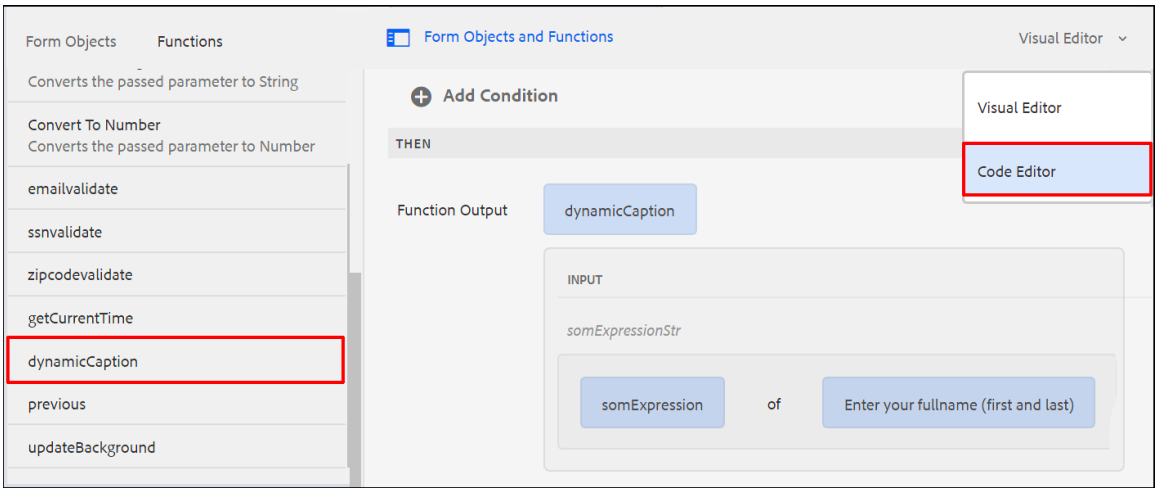

63. Click **Create**.
64. If the *Form Objects and Functions* panel is not already open, click **Form Objects and Functions**.
65. Click **Select State** in the WHEN panel and select **is initialized**.
66. Click **Select Action** in the THEN panel and select **Function Output**.
67. Select the **Functions** tab on the left to open the Functions panel.
68. Drag and drop the **dynamicCaption** function to the *Drop function or select here* box.
69. Click the *Select Option* drop-down in the INPUT panel and select **Object Property** *(see illustration)*.




70. Click *Tap here to select a property* and select **somExpression STRING**.

71. Click *Drop object or select here* and select **Enter your fullname (first and last) TEXT FIELD**.

Your condition will now look like this.



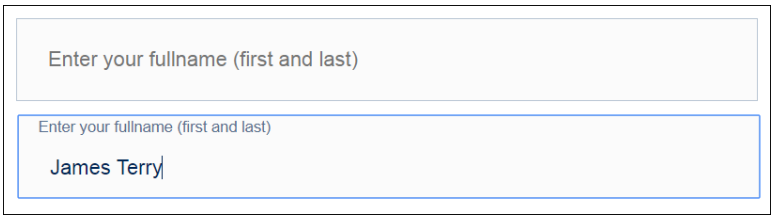

72. Switch from *Visual Editor* to *Code Editor (see illustration above)*.

This is the code created by your Visual Rule.

**dynamicCaption(this.somExpression);**

73. Click **Done** and **Close**.

74. Click **Preview** and fill in the field.




You will see the caption is dynamically resized and repositioned when you start to type.

### Explanation of the dynamicCaption function

The *dynamicCaption* function takes a string parameter. We pass it **this.somExpression** which is a reference to our control (i.e., the dynamiccaption Text Box).

**guide[0].guide1[0].guideRootPanel[0].dynamicacaption[0]**

Line 2 of the function uses the *resolveNode* method of *guideBridge* to get the handle of our control and assign it to the variable **senderCtrl**.

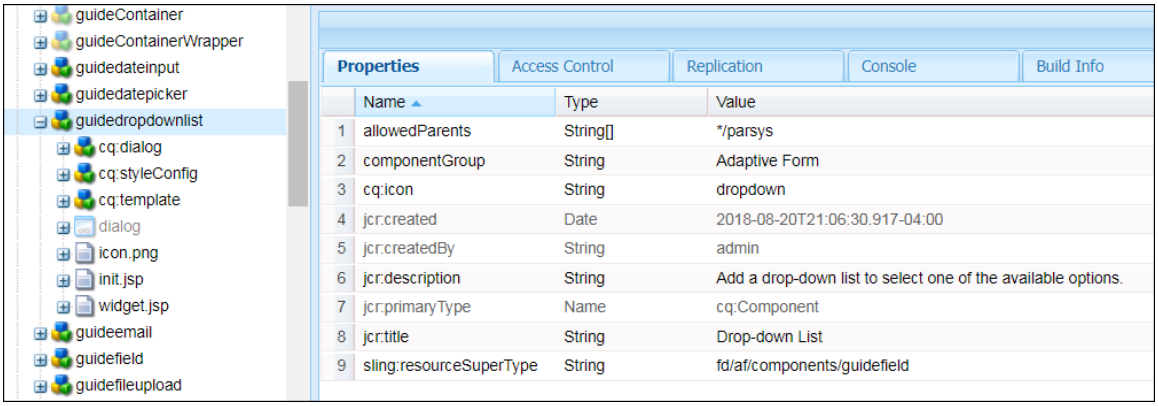
```
1  function dynamicCaption(somExpressionStr){
2      var senderCtrl = guideBridge.resolveNode(somExpressionStr);
3      var pdiv=$('#'+senderCtrl.id);
4      pdiv.addClass('dynamiccapcss');
5      pdiv.find('input:text').keyup(function () {
6          var txtctrl = $(this);
7          var curTxt = txtctrl.val();
8          var lblCtrl = txtctrl.parent().parent().find('label');
9          if (curTxt.length > 0) {
10             lblCtrl.parent().show();
11             lblCtrl.show();
12             txtctrl.css('padding-top', '18px')
13             //txtctrl.css('height', '46px')
14         }
15         else {
16             lblCtrl.parent().hide();
17             lblCtrl.hide();
18             txtctrl.css('height', '64px');
19             txtctrl.css('padding-top', '0')
20         }
21     })
22 }
```

Line 22, Column 2

## US States Drop-down List

This component will create a Drop-down List of 50 states.

1. Open **CRXDE | Lite** if it is not already opened.
2. Navigate to **/libs/fd/af/components**.
3. Select the **guidedropdownlist** node. Notice the child nodes and properties.

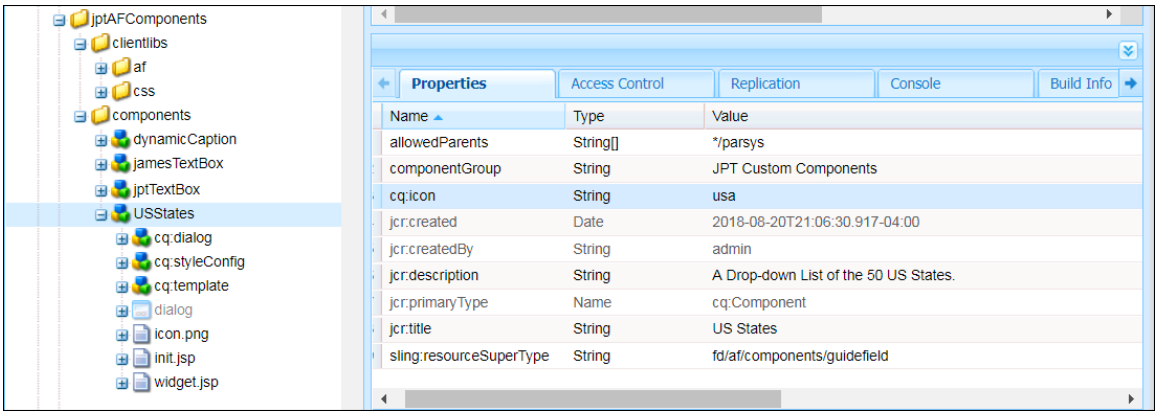



4. Right-click the **guidedropdownlist** node and select **Copy**.
5. Go back to **apps/<teamfolder>/<personalfolder>/components** and select **Paste**.
6. Click **Save All**.

7. Right-click the **guidedropdownlist** node and select **Rename**.
8. Enter **USStates** as the name.
9. Click **Save All**.
10. Make sure **USStates** is selected so you can see its properties on the right.
11. Enter **<yourname> Custom Components** as the componentGroup.
12. Enter **usa** as the cq:icon.
13. Enter **A Drop-down List of the 50 US States** as the jcr:description.
14. Enter **US States** as the jcr:title.
15. Click **Save All**.

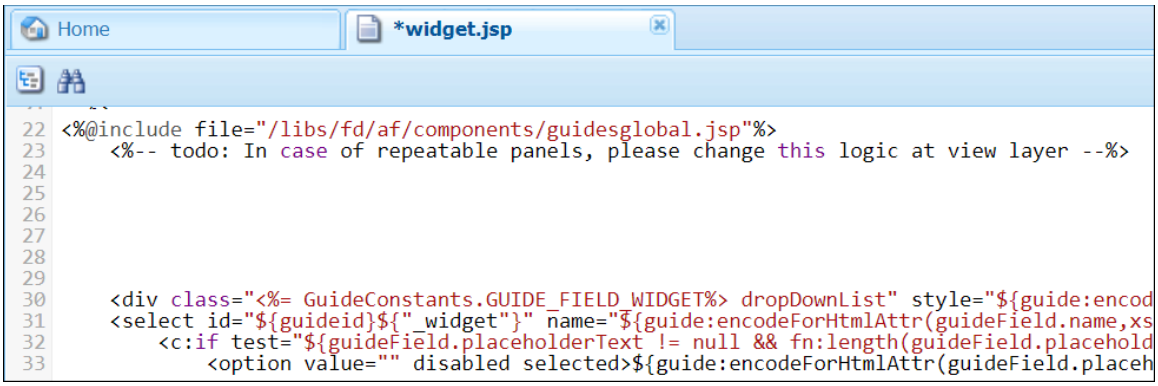
Your application should now look like this.




16. Expand the **USStates** node and select the **cq:template** child node.
17. Enter **US States** as the jcr:title.
18. Click **Save All**.

## Update the JSP

19. Expand your **USStates** component node.
20. Double-click the **widget.jsp** node.
21. Open **usstates-js-new.txt** from your Student Files in a Text Editor.
22. Go back to CRXDE and locate the **<%-- todo:** comment in the **widget.jsp** file.
23. Make some space after the **<%-- todo:** comment. Your new script tag will go here *(see illustration)*.



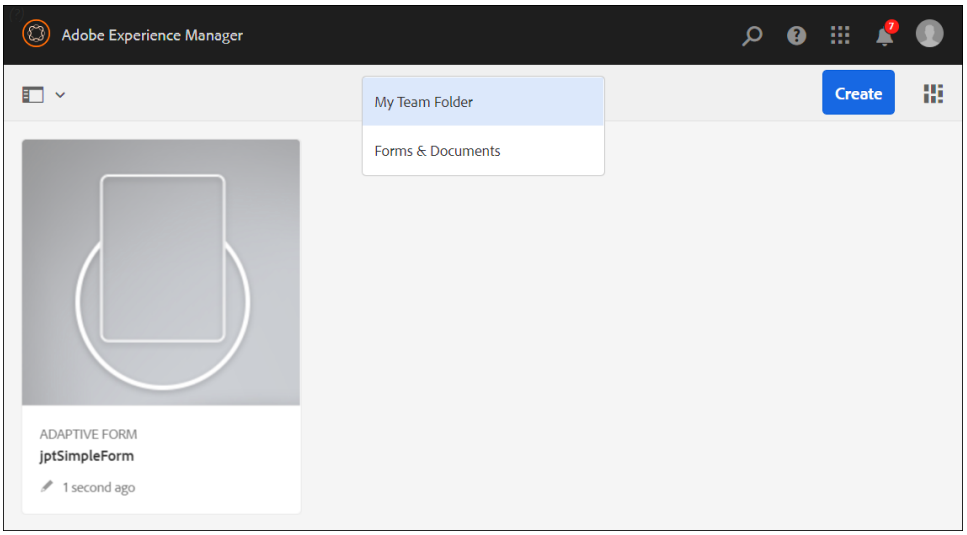

24. Copy the script tag from your **usstates-js-new.txt** file and paste it into the new white space in your widget.jsp file. Make sure to copy everything from the opening script tag to the closing script tag.

```
<script type="text/javascript">
    $(function(){
      var
stateValue=["AL","AK","AZ","AR","CA","CO","CT","DE","DC","FL","GA","HI","ID","IL","IN","IA","KS","KY","LA","ME","
MD","MA","MI","MN","MS","MT","NE","NV","NH","NM","NY","ND","OH","OK","OR","PA","RI","SC","SD","TN","TX","
UT","VT","VA","WA","WI","WY"];
      var
stateText=["Alabama","Alaska","Arizona","Arkansas","California","Colorado","Connecticut","Delaware","District of
Columbia","Florida","Georgia","Hawaii","Idaho","Illinois","Indiana","Iowa","Kansas","Kentucky","Louisiana","Maine","
Maryland","Massachusetts","Michigan","Minnesota","Mississippi","Missouri","Montana","Nebraska","Nevada","New
Hampshire","New Jersey","New Mexico","New York","North
Dakota","Ohio","Oklahoma","Oregon","Pennsylvania","Rhode Island","South Carolina","South
Dakota","Tennessee","Texas","Utah","Vermont","Virginia","Washington","Wisconsin","Wyoming"];

      var ddlusstates=$("#${guideid}${"_widget"}");
      ddlusstates.html('');

      for(var i=0; i<stateValue.length;i++){
        //$("#${guideid}${"_widget"}").append("<option value='"+stateValue[i]+"'>"+stateText[i]+"</option>");
        if(stateValue[i] == "AL"){
          ddlusstates.append('<option selected="selected" value="' + stateValue[i] + '">'  + stateText[i] + '</option>');
        }
        else{
          ddlusstates.append("<option value='"+stateValue[i]+"'>"+stateText[i]+"</option>");
        }

      }

    });

  </script>
```

25. Click **Save All**.

## Add the component to your form

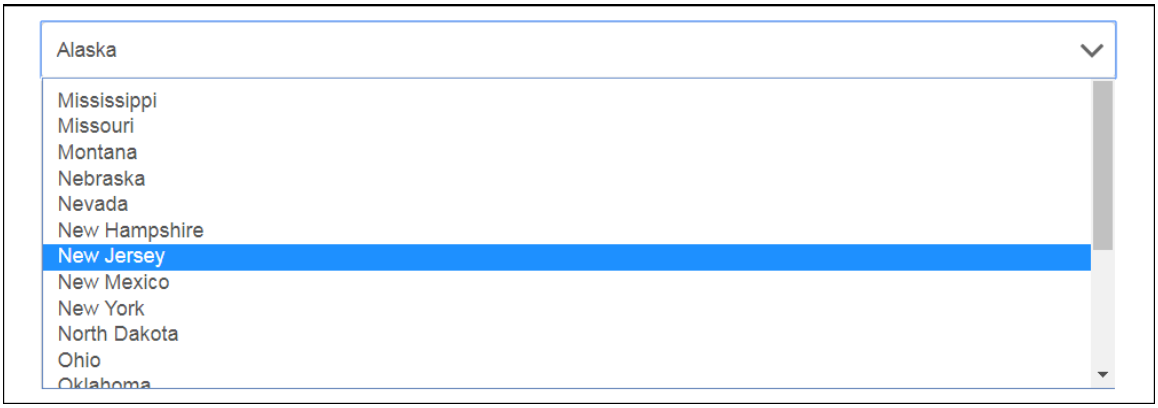
26. Go back to **AEM Forms**.




27. Open your adaptive form and make sure it is in **Edit** mode.
28. Click **Refresh** or **Reload this page** in your browser.
29. Select **Components** on the left.
30. Select **<yourname> Custom Components** to filter your components list. You should just see the components in your custom component group when this filter is applied.

    ***Note:*** *You may need to click Refresh or Reload this page to see your new component.*

31. Drag and Drop the **US States** component to your form.
32. Select the component and click **Configure** *(the wrench icon)*.
33. Change the Name to **usstates**.
34. Click **Done**.
35. Click **Preview** and click the Drop-down List. You should see all 50 states.

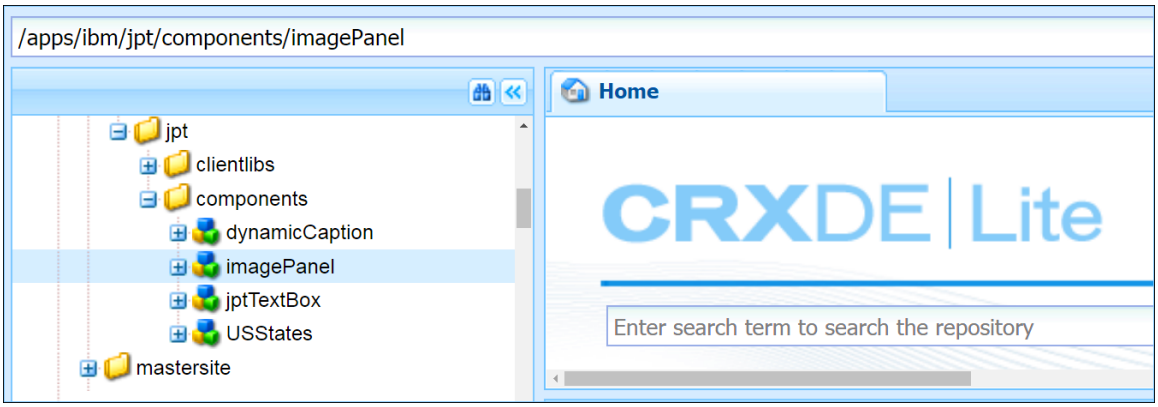



## Image Panel

This component will enable the Author to select an image for a panel background.


***Note:*** *This component works on AEM Forms on OSGi but has some issues on AEM Forms on JEE.*

1. Open **CRXDE | Lite** if it is not already opened.
2. Navigate to **/libs/fd/af/components**.
3. Locate the **guideimage** node.
4. Right-click the **guideimage** node and select **Copy**.
5. Go back to **apps/<teamfolder>/<personalfolder>/components** and select **Paste**.
6. Click **Save All**.
7. Right-click the **guideimage** node and select **Rename**.
8. Enter **imagePanel** as the name.

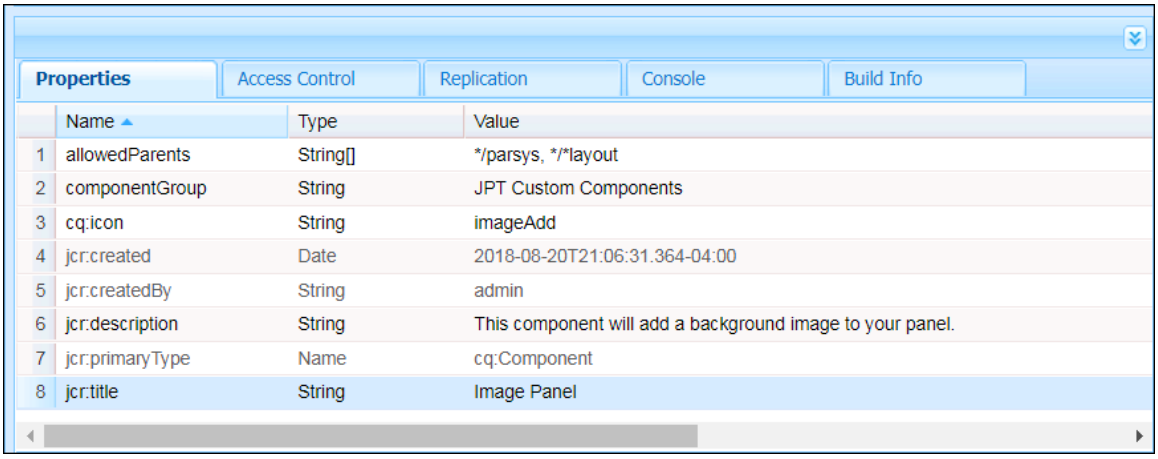



9. Click **Save All**.
10. Make sure **imagePanel** is selected so you can see its properties on the right.
11. Enter **<yourname> Custom Components** as the componentGroup.
12. Enter **imageAdd** as the cq:icon.
13. Enter **This component will add a background image to your panel** as the jcr:description.
14. Enter **Image Panel** as the jcr:title.

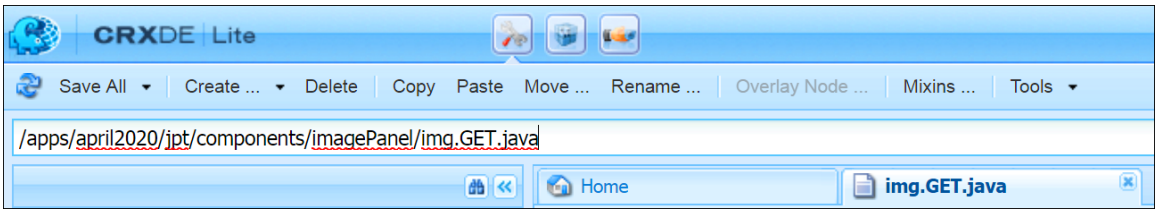
15. Click **Save All**.

Your application should now look like this.




16. Select the **cq:template** child node under the **imagePanel** node.
17. Enter **Image Panel** as the jcr:title.
18. Click **Save All**.
19. Double-click the **img.GET.java** file to view the code.
20. Update the package definition in the Java file so it references your component path. This is the first line after the comment block.

    ***Note:*** *The values in the CRX path will help you make your package definition. Here is one example. Your details will be different but notice how I replaced the slashed with dots in this reference below.*




If your path looks like the one above, your package reference will look like this.

```
package apps.april2020.jpt.components.imagePanel;
```

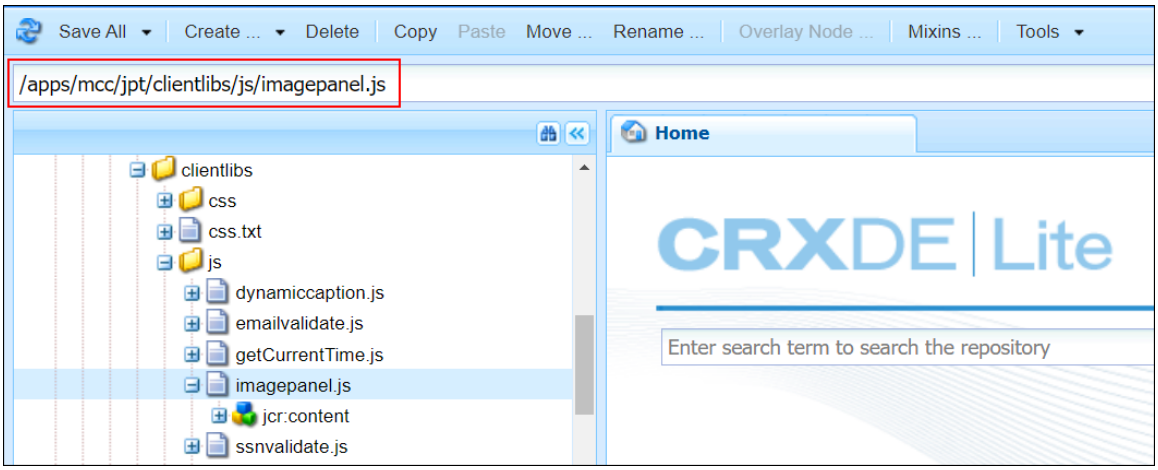
21. Click **Save All**.
22. Expand your client library folder.

**/apps/<teamfolder>/<personalfolder>/clientlibs**

23. Right-click on the **js** folder and select **Create – Create File**.
24. Enter **imagepanel.js** for the *Name* and click **OK**.
25. Click **Save All**.

26. Expand **imagepanel.js** so you can select its **jcr:content** node.
27. Double-click the **jcr:data** property.
28. Click **Browse** and locate the **imagepanel-js.txt** file in your Student Files.
29. Select the file and click **Open**.
30. Click **OK**. You should now see the JavaScript that has been added to your node *(see illustration)*.


31. Click **Save All**.
32. Copy the reference to **imagepanel.js** and save it in NotePad or a similar Text Editor. We will use this reference in a future step.




33. Expand your components folder.
34. Right-click on **guideimage.jsp** and select Rename.
35. Enter **imagePanel.jsp** for the node name.
36. Click **Save All**.
37. Expand **imagePanel.jsp** so you can select its **jcr:content** node.

38. Double-click the **jcr:data** property.

39. Click **Browse** and locate **imagepanel.jsp** in your Student Files.

40. Select **imagepanel.jsp** and click **Open**.

41. Click **OK**.

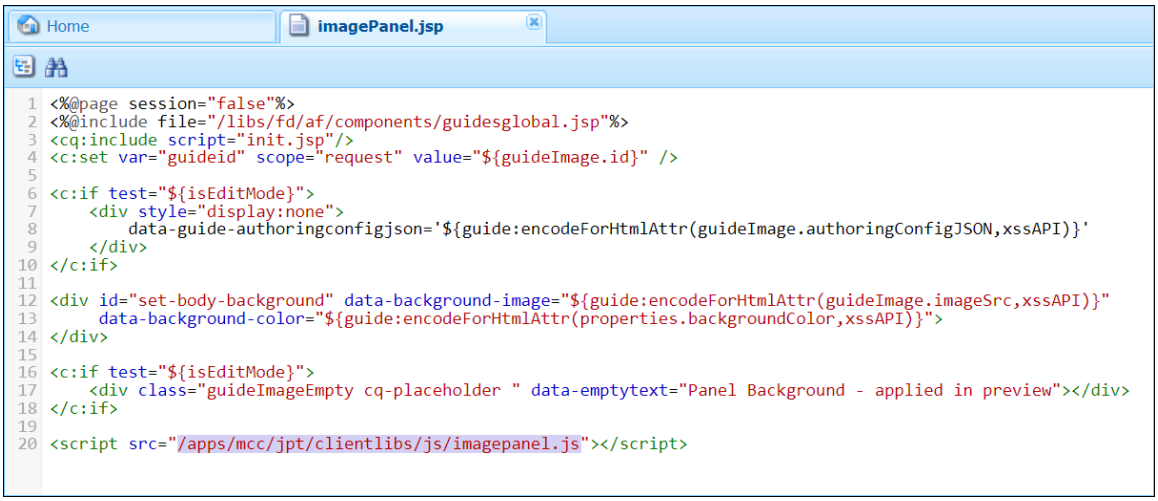
42. Click **Save All**.

43. Double-click on the **imagePanel.jsp** node.

44. Update this reference at the bottom of the file so it points to your JS file. This is the reference you pasted into Notepad.

**<script src="/apps/<teamfolder>/<personalfolder>/clientlibs/js/imagepanel.js"></script>**

45. Your imagePanel.jsp code should now look like this.




46. Click **Save All**.

47. Update your **js.txt** file with a reference to your new file.

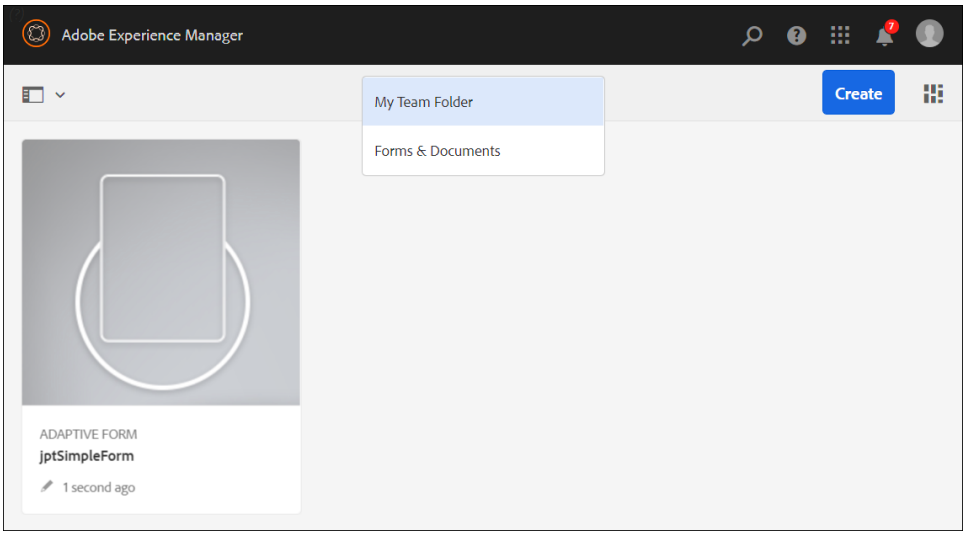
**#base=js**
**imagepanel.js**

> ***Note:*** *We do not really need to updated our js.txt file to point to imagepanel.js. We added a script tag to our JSP file and this will work without adding this reference in the js.txt file.*

48. Click **Save All**.

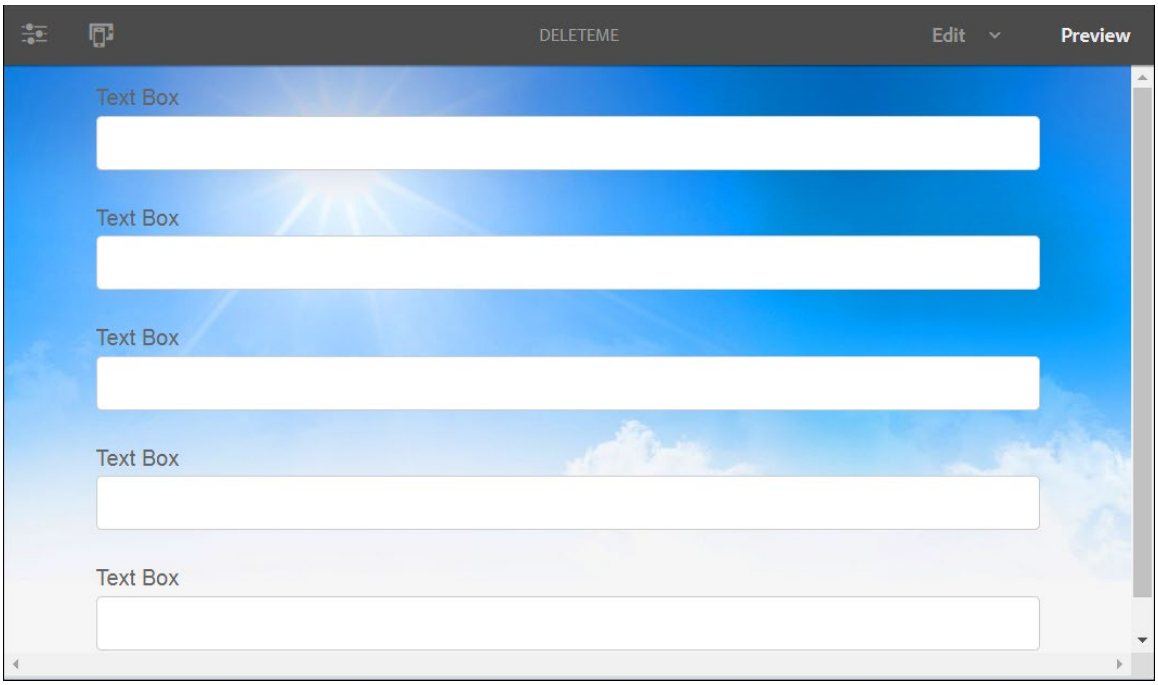
## Add the component to your form

49. Go back to **AEM Forms**.



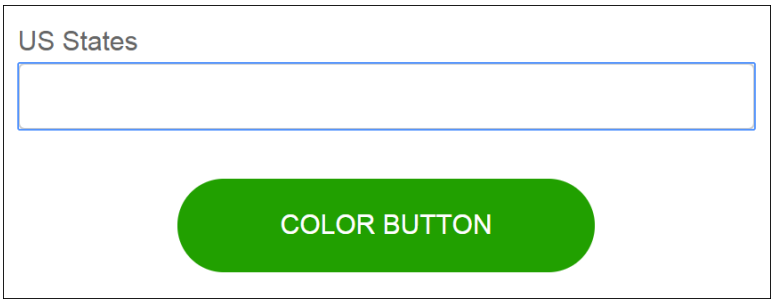

50. Open your adaptive form and make sure it is in **Edit** mode.
51. Click **Refresh** or **Reload this page** in your browser.
52. Select **Components** on the left.
53. Select **<yourname> Custom Components** to filter your components list. You should just see the components in your custom component group when this filter is applied.
54. Drag and Drop the **Image Panel** component to your form.
55. Select the component and click **Configure** *(the wrench icon)*.
56. Enter **myImagePanel** as the *Name*.
57. Click **Upload**.
58. Select the **clear-blue-sky.jpg** file in your Student Files and click **Open**.
59. Click **Done**.
60. If you don't already have some fields on your form then drag and drop some Text Boxes to your form.
61. Click **Preview**.

You will see the image filling the panel background.




## Color Button

This component will add additional types to the standard button that enable the Author to select a button color.



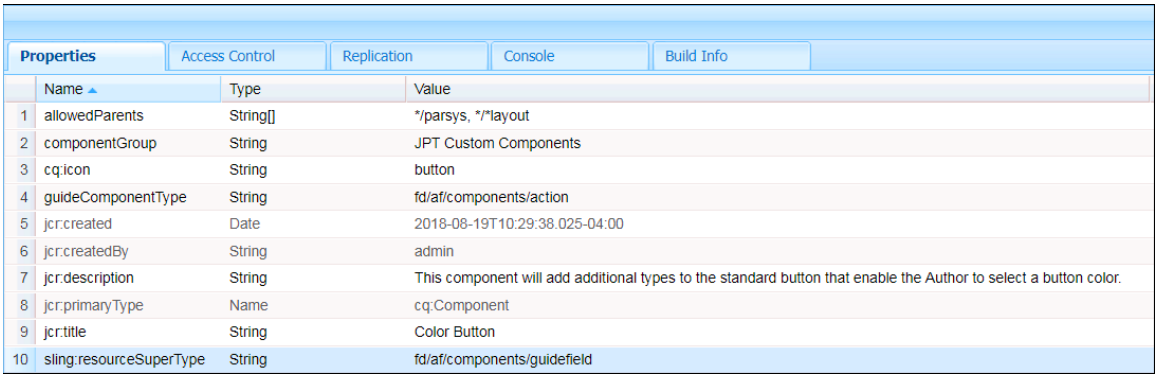

1. Open **CRXDE | Lite** if it is not already opened.
2. Navigate to **/libs/fd/af/components**.
3. Right-click the **guidebutton** node and select **Copy**.
4. Go back to **apps/<teamfolder>/<personalfolder>/components** and select **Paste**.
5. Click **Save All**.
6. Right-click the **guidebutton** node and select **Rename**.
7. Enter **colorButton** as the name.
8. Click **Save All**.
9. Make sure **colorButton** is selected so you can see its properties on the right.
10. Enter **<yourname> Custom Components** as the componentGroup.

11. Enter **This component will add additional types to the standard button that enable the Author to select a button color** as the jcr:description.

12. Enter **Color Button** as the jcr:title.

13. Click **Save All**.

Your application should now look like this.



Properties | Access Control | Replication | Console | Build Info

| | Name | Type | Value |
|---|---|---|---|
| 1 | allowedParents | String[] | */parsys, */*layout |
| 2 | componentGroup | String | JPT Custom Components |
| 3 | cq:icon | String | button |
| 4 | guideComponentType | String | fd/af/components/action |
| 5 | jcr:created | Date | 2018-08-19T10:29:38.025-04:00 |
| 6 | jcr:createdBy | String | admin |
| 7 | jcr:description | String | This component will add additional types to the standard button that enable the Author to select a button color. |
| 8 | jcr:primaryType | Name | cq:Component |
| 9 | jcr:title | String | Color Button |
| 10 | sling:resourceSuperType | String | fd/af/components/guidefield |

14. Select the **cq:template** child node under the **colorButton** node.

15. Enter **Color Button** as the jcr:title.

16. Click **Save All**.

17. Double-click **widget.jsp** to open the JSP code.

18. Create some space before the first ** **.

19. Add this taglib and tag in the space you created *(see illustration)*.

**<%@taglib prefix="ui" uri="http://www.adobe.com/taglibs/granite/ui/1.0" %>**
**<ui:includeClientLib categories="<your client Library category>"/>**

20. Click **Save All**.

21. Replace **"<your client Library category>"** with the name of your client library.

Your code should look like this.

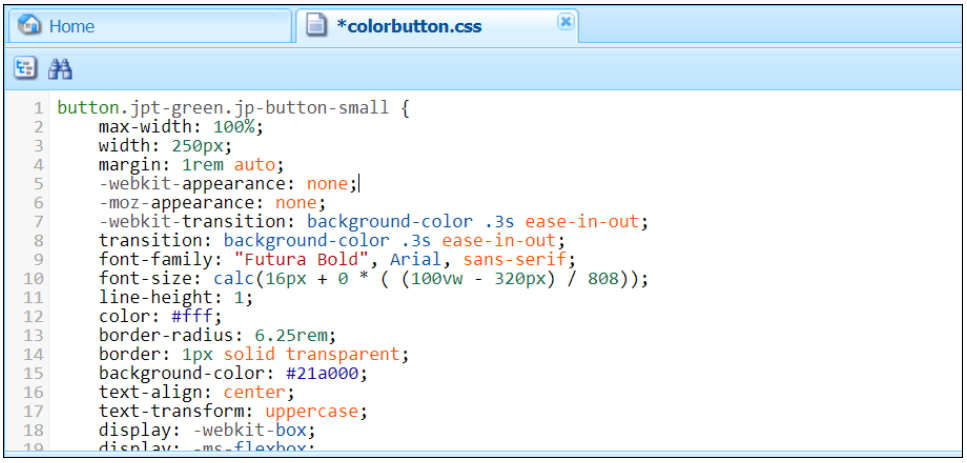
```
<%@ page import="com.adobe.aemds.guide.utils.GuideConstants" %>
<%--
  Button Component
--%>
<%@include file="/libs/fd/af/components/guidesglobal.jsp" %>
<%GuideNode guideField =(GuideNode) request.getAttribute("guideField");%>

    <%@taglib prefix="ui" uri="http://www.adobe.com/taglibs/granite/ui/1.0" %>
    <ui:includeClientLib categories="af.jptcomponents"/>

<div class="<%= GuideConstants.GUIDE_FIELD_WIDGET%> <%= GuideConstants.GUIDE_FIELD_BUTTON_WIDGET%>" style="${guide:encodeForHtmlAttr(guideFi
    <button class="${guideField.buttonType} ${guideField.buttonSize} ${guideField.type}" type= ${guideField.type == "submit" ? "submit" : "b
        <span class="iconButton-icon"></span>
        <span class="iconButton-label" data-guide-button-label="true" style="${guide:encodeForHtmlAttr(guideField.captionInlineStyles,xssAPI
    </button>
    <%-- End of Widget Div --%>
</div>
```

22. Click **Save All**.

## Add the CSS to the Client Library

23. Right-click on the **css** folder and select **Create – Create File**.
24. Enter **colorbutton.css** for the Name and click **OK**.
25. Click **Save All**.
26. Expand **colorbutton.css** so you can select its **jcr:content** node.
27. Double-click the **jcr:data** property.
28. Click **Browse** and navigate to the **colorButton-css.txt** file in your Student Files.
29. Select the file and click **Open** and you will see the file reference in your dialog box.
30. Click **OK** and the CSS will load into the node.

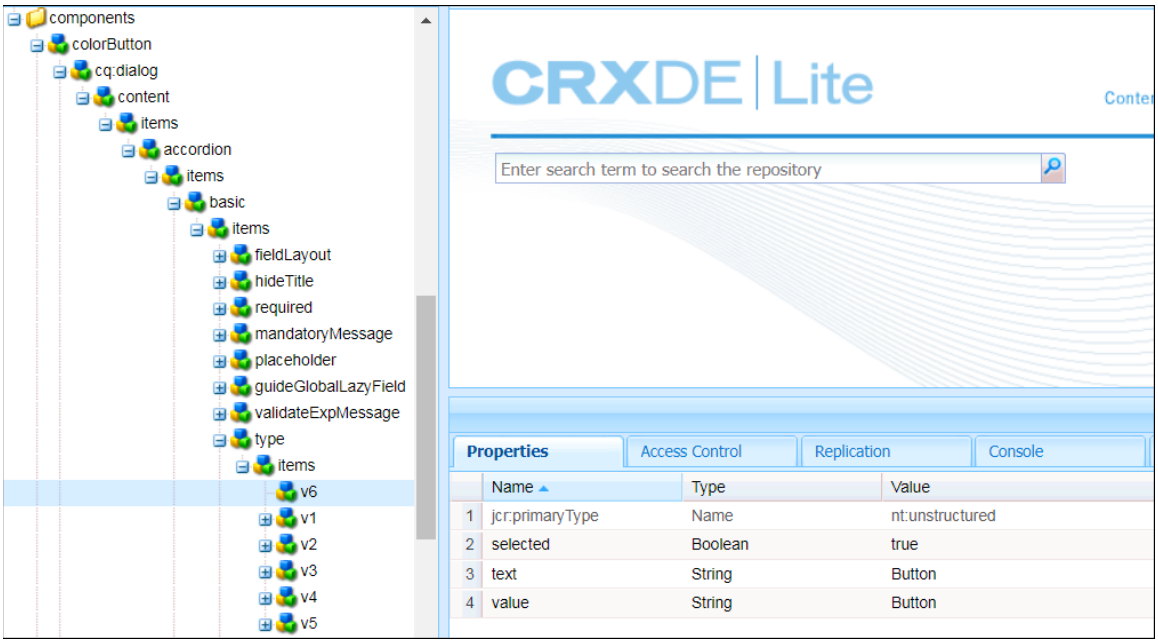



31. Click **Save All**.
32. Update your **css.txt** file with a reference to your new file.

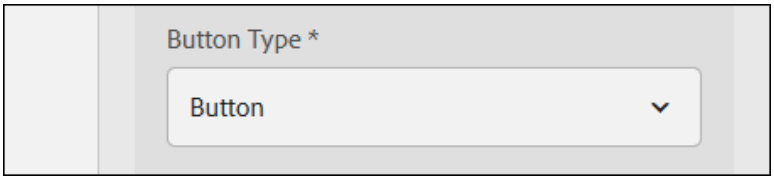
**#base=css**
**colorbutton.css**

33. Click **Save All**.
34. Go back to your **colorButton** component and expand the **cq:dialog** node.
35. Expand the child nodes until you get to this node *(see illustration)*.

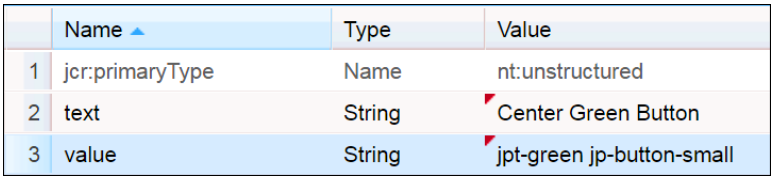
**/cq:dialog/content/items/accordion/items/basic/items/type/items**




36. Notice that there are 6 types. Each of these types is available to the Form Author.




37. Right-click the **v1** node and click **Copy**.
38. **Paste** the node under items three times so you have three new nodes.
39. Click **Save All**.
40. Right-click on each new node and select **Rename** and update the names to be **v7**, **v8**, and v9.
41. Click **Save All**.
42. Select the **v7** node so you can see its properties on the right.
43. Enter **Center Green Button** for the Value of your text property.
44. Enter **jpt-green jp-button-small** for the Value of your value property. *Note:* This is a selector in the colorbutton.css file *(see illustration)*.



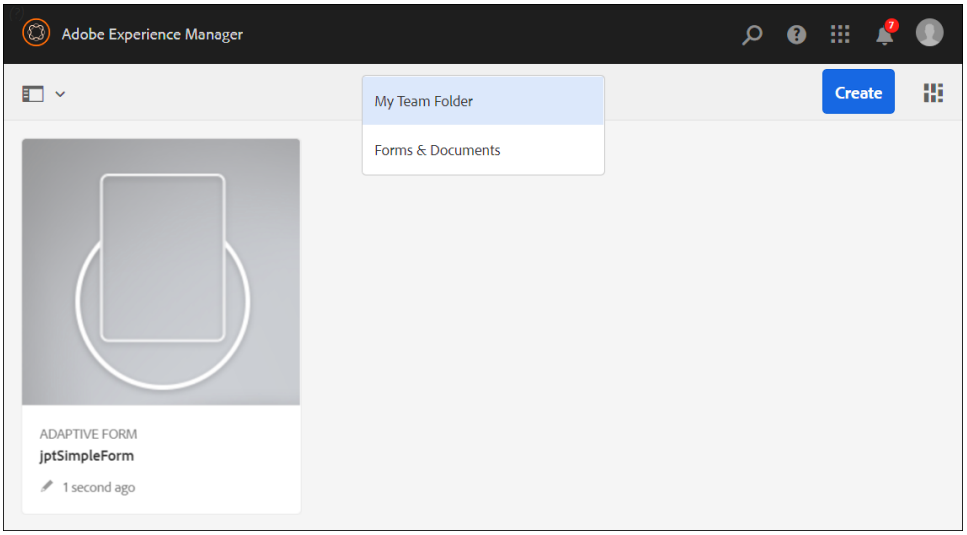

45. Click **Save All**.
46. Select the **v8** node so you can see its properties on the right.

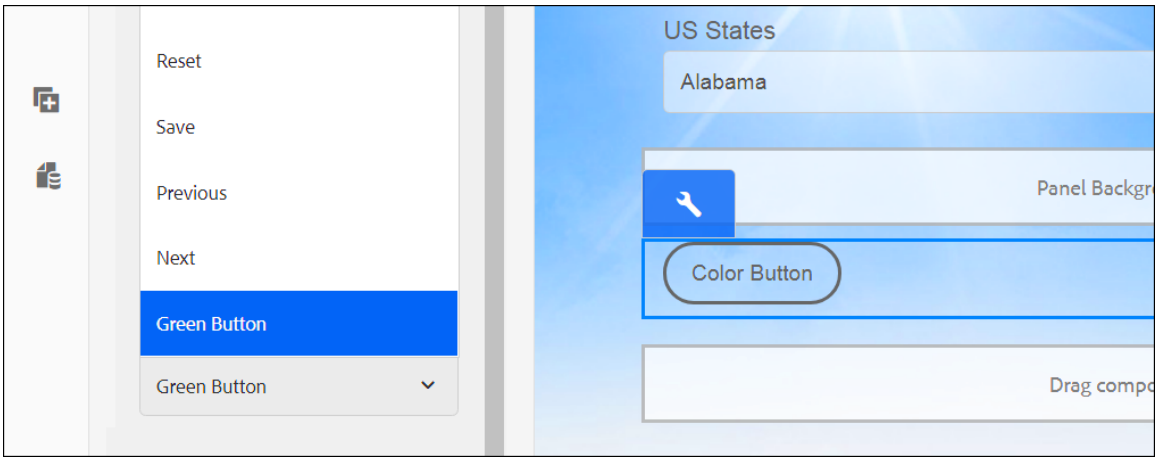
47. Enter **Center Yellow Button** for the Value of your text property.
48. Enter **jpt-primary-button jp-button-instance** for the Value of your value property. *Note:* This is a selector in the colorbutton.css file.
49. Click **Save All**.
50. Select the **v9** node so you can see its properties on the right.
51. Enter **Hyperlink** for the Value of your text property.
52. Enter **button-hyperlink jp-button-hyperlink** for the Value of your value property. *Note:* This is a selector in the colorbutton.css file.
53. Click **Save All**.

## Add the component to your form

54. Go back to **AEM Forms**.



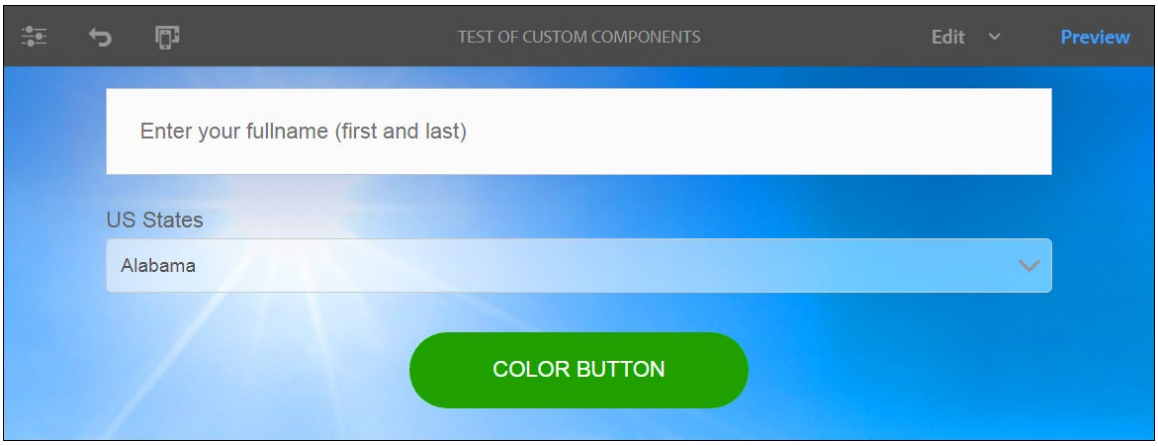

55. Open your adaptive form and make sure it is in **Edit** mode.
56. Click **Refresh** or **Reload this page** in your browser.
57. Select **Components** on the left.
58. Select **<yourname> Custom Components** to filter your components list. You should just see the components in your custom component group when this filter is applied.
59. Drag and Drop the **Color Button** component to your form.
60. Select the component and click **Configure** *(the wrench icon)*.
61. Change the name to simply **colorbutton**.
62. Click the **Button Type** drop-down and select **Center Green Button**.




63. Click **Done**.

64. Click **Preview**.

You will see your new Center Green Button.




Experiment with some additional button types that point to different CSS selectors.

# About this Courseware

SmartDoc Technologies supplied the official *Adobe-Certified* AEM Forms Training courseware to Adobe and Adobe's clients from 2016 – 2021. During that time, our SmartDoc Courseware was battle-tested by thousands of students worldwide. Our SmartDoc Technologies courseware has been peer-reviewed and certified by the Adobe *Engineering*, *Product*, and *Curriculum* teams and by thousands of students like you.

In addition to having the highest *quality* AEM Forms courseware, the SmartDoc library also has the highest *quantity* of AEM Forms courseware. You will find the perfect course for your specific AEM Forms needs in the SmartDoc library. You can always find a current listing of our Adobe AEM Forms training courses on our website.

**www.smartdoctech.com**