Table of Contents

[Page Edit details: 2](#_Toc349320604)

[Gadget Tag 3](#_Toc349320605)

[Sample structure of a gadget tag in the page: 3](#_Toc349320606)

[Edit Page structure: 3](#_Toc349320607)

[Page Layout diagram: 4](#_Toc349320608)

[Files for loading and editing of SCO pages: 4](#_Toc349320609)

[Generating Gadget Forms 4](#_Toc349320610)

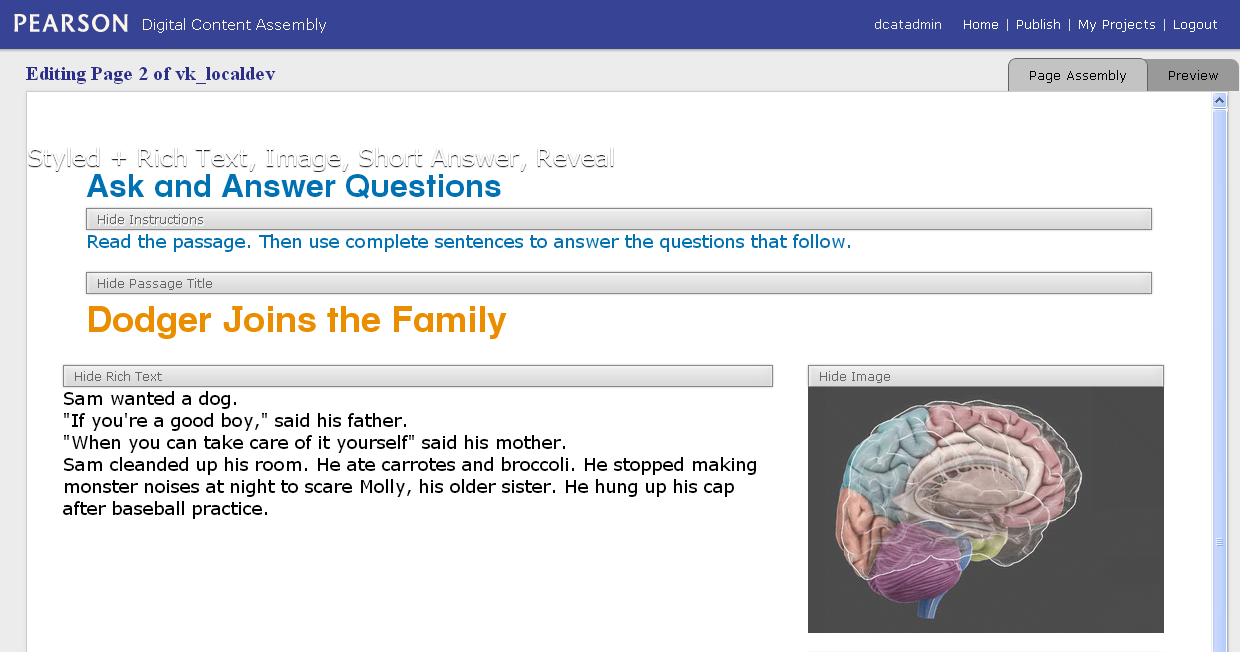
[Sequences for generating a gadget form: 5](#_Toc349320611)

Gadget

## Page Edit details:

A SCO page is loaded in “Page Edit” page Gadget loaded in Page Edit page:

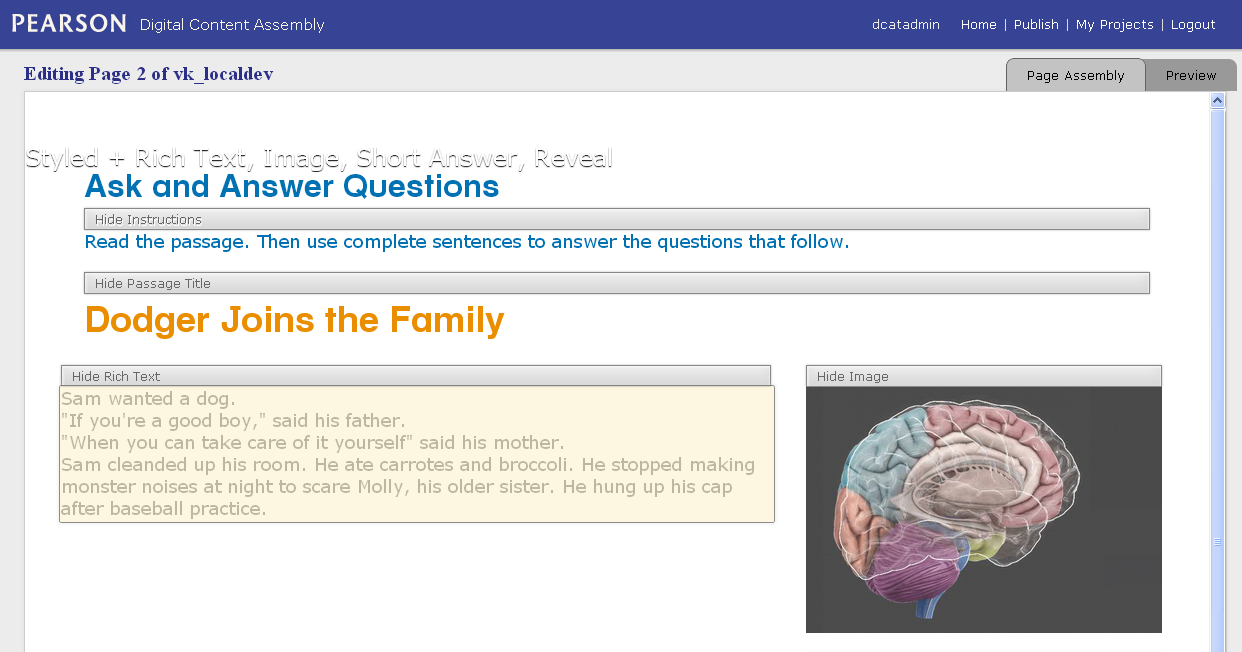
Tabs for switching between **Page Assembly** and **Preview** modes



**Rich Text** Gadget

**Image** Gadget

A **Show**/**Hide** button appears in **Page Assembly** mode



Rolling over any gadget will highlight the gadget area with a transparent yellow box.

## Gadget Tag

### Sample structure of a gadget tag in the page:

### <div class="*image* gadget *identifier-image*" data-id="*g4*" data-gadget="*Image*" data-collapsible="*Image*" style="*min-width: 20px; min-height: 20px;*">

...

</div>

#### Attributes and values:

* **class**:
  + ***image***: Specific class for the gadget type. It “**image**” for “**Image**” gadget, “**richtext**” for “Rich Text” gadget and so on.
  + **gadget**: this value is common to all gadget to identify <div></div> tags for gadgets to apply common
  + ***others***: any other classes
* **data-id**: id of data for this gadget in configuration
* **data-gadget**: values of this attribute is used identify the type of gadget in script and apply configuration accordingly. Existence of this attribute is determined before adding “**mouseover”** event to highlight the area of this <div></div> tag by drawing a rectangle over it.
* **style** : for inline styles
* **data-collapsible**: If some value is assigned to this attribute, it will include a Show/Hide button to at the top of the gadget in the page. The value will be displayed as label with prefix Show/Hide for the button.

## Edit Page structure:

Here we will discuss the structure of the page which load a page of a SCO at a time and provide controls to edit the content of it. Edit page (app\views\Page\edit.html) provide top level of UI and controls and functionality like tab control for switching between Assembly mode and Preview mode, generating Gadget editor forms and submitting modified data. There a <iframe> tag in this page which loads app\views\Page\loadPageForEditing.html page which actually loads and displays content of a SCO page to be edited.

### Page Layout diagram:

Page Edit page (app\views\Page\edit.html)

javascripts/page/edit.js

javascripts/edit/edit-interface.js

javascripts/gadget/gadget-config.js

javascripts/gadget/gadget-asset-manager.js

javascripts/edit/edit-interface.js

<iframe></ iframe>

(app\views\Page\loadPageForEditing.html)

### Files for loading and editing of SCO pages:

#### app\views\Page\edit.html

* *javascripts/page/edit.js:* Provides interaction with top level controls like switching between Assembly mode and Preview mode, back and continue.
* *javascripts/edit/edit-interface.j:* Interacts with gadget page, control middle level interaction like highlighting gadget area on mouseover, display gadget form on click, and submit modified data for the SCO page.
* *javascripts/gadget/gadget-config.js:* Responsible for creating and controlling Gadget editor forms (Panels).
* *javascripts/gadget/gadget-asset-manager.js*: Responsible for creating assets of gadget and manage it.

#### app\views\Page\loadPageForEditing.html

* + *javascripts/edit/edit-interface.js:* Responsible for loading data of a SCO page and generate the page, configuring and reconfiguring all gadgets with data, providing low level interface for gadget editing like generating and controlling show/hide button for gadgets, turn on/off edit mode, and adding interactivity of mouse with gadget in edit mode.

## Generating Gadget Forms

Each gadget has its own structure, functionalities, and dataset so the structure of editor form depends on that. To define the structure of each gadget form DCAT used xml schema document. There is a schema xml document defined for each gadget form which is used further to generate the form. As we click on a gadget in Assembly mode DCAT detects type of the gadget, loads schema for it (if not loaded already), uses it to generate the editor form, and displays within a popup panel.

### Sequences for generating a gadget form:

#### Detecting “mouseover” state with a gadget in the SCO page: - edit-player.js

**Code**: **Com.Edit.Player.edit()**  
This method is responsible for detecting the mouseover state of the Gadget and collects config and required information from it and passes to Com.Edit.Interface.edit - *edit-interface.js*for highlighting the gadget.

#### Highlighting Gadget: - edit-interface.js

**Code**: **Com.Edit.Interface.edit(id, type, rectangle, config)**  
 Takes parameters from Com.Edit.Player.edit - *edit-player.js*, and crate highlight area for the gadget, and prepare for handling click on the highlighted area.

#### Click on the gadget: - edit-interface.js

**Code**: **Com.Edit.Interface.SELECTOR.bind("click", function(){...})**

#### Load schema document: - edit-interface.js

**Code**:

**$.ajax({**

**type: "GET",**

**url: Com.Edit.Interface.GET\_SCHEMA\_URL(type), // baseUrl + "/" + Com.Edit.Interface.SCHEMA\_DIRECTORY + "/" + type.toLowerCase() + "." + "xml",**

**dataType: "xml",**

**success: function(schema) {**

**...**

**Com.Gadget.Config.show(schema, config);**

**...**

**}**

**})**

#### Generating and Showing Gadget form: - gadget-config.js

**Code:**   
**Com.Gadget.Config.show(schema, config)**  
Takes parameters *edit-interface.js* once schema document is loaded, generate gadget form based on schema provided for the gadget and configure it with the saved values for the gadget. It parses the schema, crates components of the form at runtime based on the type of nodes and its attribute values in the schema. There is a separate detailed document about the schema and various elements of it. You can also visit [here](http://dcatppe.pearson.com/documentation/config) for the documentation of the schema elements.