Cartridge Builder

Requirements

- Jive Anywhere 2.3 or higher
- Internet Explorer 9 or higher, Chrome, Firefox or Safari running on Windows or Mac OS X
- Websites running in compatibility mode are not supported on Internet Explorer
- Basic knowledge of HTML is recommended

Overview

A cartridge is a plug-in that extends the functionality of Jive Anywhere for specific web-pages or web-apps. The most popular functionality of a cartridge is to generate a custom preview section that is rendered into every discussion created for a web page. For instance, you might create a cartridge for CNN.com so every discussion created on that website will include an automatic preview of the article.

Cartridges are natively written in JavaScript and require some developer knowledge. The Cartridge Builder is a UI tool designed to make it easier to create basic and even complex cartridges for Jive Anywhere without a single line of code. However, in order to achieve that you have to "teach" this tool how to extract the data from the web page and render the preview section accordingly.

After you've created your cartridge you can test it, upload it to the Jive server and even share it across all users of the community (requires administrator privileges).

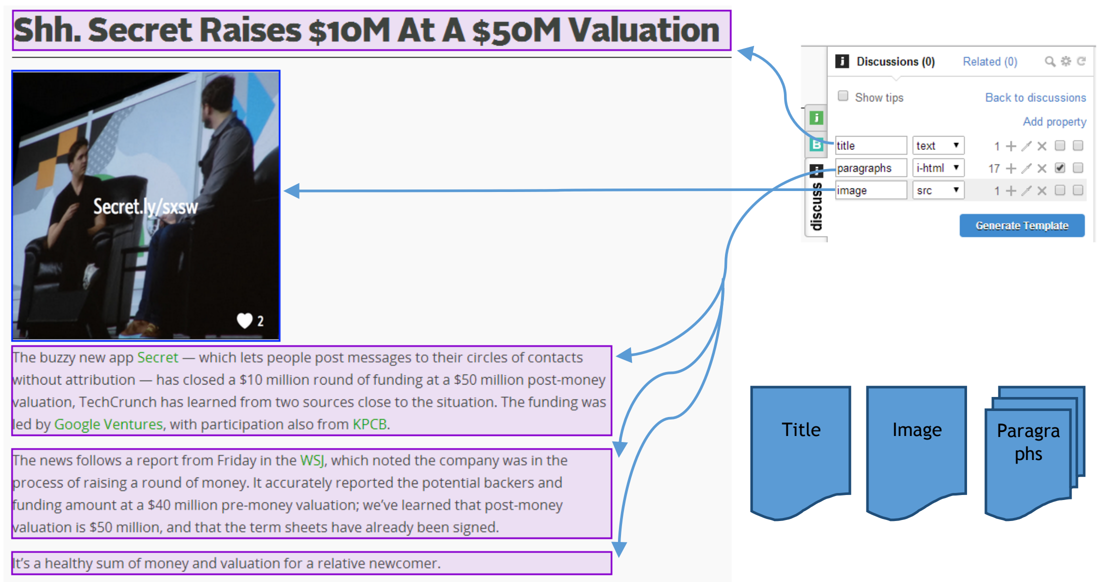
Choosing a website for your cartridge

This is the first step of creating a cartridge. You need to decide what kind of webpages are going to be supported by your cartridge. The web pages have to contain a similar visual tree so the same extraction logic could be used for all of them. For instance you might choose an article of a news site, a thread page on a forum or a document on your internal corporate web app. If you need to support more than one type of content with a different visual tree you might end up by writing multiple separated cartridges. It is also important to being able to determine which web pages are about to activate your cartridge by evaluating their URLs – but this is a step for a later stage, for now just load the web page you wish to work with for your cartridge into your browser. For this demonstration we have chosen an article page in TechCrunch.

After the page is loaded, click on the Jive Anywhere notification button to open its sidebar. In case a cartridge already exists for this web page and is available for your account, you will see a **"[cartridge name] cartridge is active"** notification at the top of the sidebar. If a cartridge could not be found you will see "There are no cartridges for this page, let me create one" notification with a link. Click on that link to start the Cartridge Builder. If the link doesn't appear or a cartridge was already created then click on the gear icon and then click Cartridge Builder.

Extracting Data

This is the most important step of creating a cartridge. Hear you're going to define how to extract the data from the visual tree of the web page. For instance, the title of an article might be easy for your eyes to identify by its location and font size, but strict rules are required in order to create reliable automated functionality across similar web pages which are going to be handled by your cartridge.



Shh. Secret Raises $10M At A $50M Valuation

The buzzy new app Secret — which lets people post messages to their circles of contacts without attribution — has closed a $10 million round of funding at a $50 million post-money valuation, TechCrunch has learned from two sources close to the situation. The funding was led by Google Ventures, with participation also from KPCB.

The news follows a report from Friday in the WSJ, which noted the company was in the process of raising a round of money. It accurately reported the potential backers and funding amount at a $40 million pre-money valuation; we've learned that post-money valuation is $50 million, and that the term sheets have already been signed.

It's a healthy sum of money and valuation for a relative newcomer.

title	text	1
paragraphs	i-html	17
image	src	1

The figure shows the flow of data extraction

Picking elements

The Cartridge Builder displays a list of properties which can be added or removed. By default 3 properties has already been created for you to play with. Each property defines a rule for selecting the DOM elements from the visual tree of the web page using jQuery selectors. You don't have to understand the syntax of jQuery selectors but it can greatly help in complex scenarios.

Let's start with the title of the article. Click the eyedropper tool of the first property and the sidebar will slide out of the screen to make more screen space available. Now try to mouse-over elements on the page and see how they are being highlighted with dashed blue lines. Move the pointer to the title of the article and make sure it's the only element highlighted. You can check the number of the elements by looking at the blue tooltip that pops up near the label "Items". When you're ready press Enter. You may also click the mouse, however it is better to avoid this behavior since a clickable hyperlink might be activated and get you out of the current page.

The element has become highlighted in purple and the sidebar slides back - showing you the number 1 at the first property row. Behind the scenes a jQuery selector was created for you and is expected to select the same element (although with different content) on other articles. If the number is different than 1 you should repeat the previous step by clicking on the eyedropper tool again. When finished click on the textbox of the first row and give the property a name by typing **title**.

You may now repeat this process by clicking the eyedropper tool of the second property and choosing the image element of the article this time. Give the property the name **image**. By default the textual data is extracted from the selected elements, but in this case we would like to extract the image source out of the element. To change the **extraction method**, click on the dropdown button that says text at the same row and select **src instead**.

Note that at any time you can just mouse-over a property row on the sidebar in order to highlight the selected elements connected to it. The elements will stay highlighted so you can scroll the page and see the entire view.

In order to check your work in progress click on the Generate Template button. We will get more depth into this on the Rendering Templates section but for now just take a look at the result. You should see a hyperlink automatically created from the page title, a brief description taken from OpenGraph (if available) and below that you'll see the title and the image you've extracted from the page in this step.

Picking list elements (arrays)

In many cases you want to extract a list of items from the web page, such as paragraphs, comments or images.

Click the eyedropper tool of the third property and mouse over one of the article's paragraphs. You should see all other paragraphs being highlighted as well and the Items label at the tooltip should display the number of paragraphs of the article. You may press the space key in order to pause the mouse hovering functionality and scroll the page up and down to see all highlighted elements. When you're ready to continue just press Enter to select the highlighted elements or press the space key to un-pause and refine your selection.

After the sidebar slides back you will see the counter label of the third property shows the number of selected paragraphs. It is important to understand this number is for reference only and is not part of the selector. It doesn't guarantee that the same number of items will be found on another article if at all.

Give the property the name **paragraphs**. Since this property selects a list of items, the array checkbox (the left one of the two) of this property should be checked. Otherwise, only the first element found by the selector will be extracted.

As before, try clicking the Generate Template button and you should now see the list of paragraphs taken from the article.

Grouped properties

Grouped properties allow you to group several properties under the same parent in hierarchal layout. This is very useful for extracting list of elements (array property) where each element contain several properties that belong to the same item.

One of the advantages of this method is that properties are strictly bounded to each other.

For this demonstration let’s open Google Search and type any text phrase to search. The webpage is displaying a list of search results, and each of them contains a title and a short description. Instead of creating one array property for all the titles and another one for all the descriptions you should create just one array property representing the container element of each search result (the title and the description). Then, under the array property of the container you should create two child properties for the title and the description.

To create a child property click the plus button at the row represents the parent. Picking the elements is almost the same as before, but this time you’ll see green boxes representing the elements of the parent property you are working on. You may notice that you can only select elements inside those green boxes since the parent element (the container) is the boundary of your selection. Notice that after adding child properties, the extraction method dropdown of the parent property has disappeared. This is done since that property is only a container for other properties and it can’t return any value by itself.

After adding the child properties and selecting the elements, click the Generate Template button and you should see a list of search results. Each item should display the title and the description below.

Tip: Multiple levels of groups can be created as well for more complex scenarios.

Self-selector properties

In some cases you want to extract data from parent properties. However a parent property doesn’t have an extraction method since it’s only used as a container for other properties.

Let’s assume we have a list of hyperlinks and we want to extract both the text and the URL of each (grouped properties). In the usual scenario we would first create a parent property for the item container such as a DIV element and mark it as array. Then we would create two child properties, for the text and the URL. But what if the container element is also the context of the extraction as in this case, the hyperlink is the container (since we couldn’t find another one) and it’s also the context that holds the title and URL.

The solution would be to use self-selector properties. After creating the hyperlink element and mark it as array, we should create two child properties. One is for the text and the other for the URL. But this time we won’t select any elements for those child properties and there will be no counter displayed for them. Each of these properties represents the same elements selected by their parent, so they are grouped together and can extract data using the context as their parent. By selecting the text extraction method for the text property and the href extraction method for the url property we can extract the data we need.

Self-selector properties can’t have child properties.

Element subtraction

This feature allows subtracting several elements from the parent’s selector. For instance, if you want to select the text of a paragraph element you can’t select the text itself (because text is not an element). Instead, you’re selecting the paragraph element including all its content. But in many cases the paragraph might contain some unwanted elements such as images or other text surround by SPAN elements.

The solution would be to use element subtraction. After creating a property for the paragraph and selecting its elements, check the Subtract checkbox (the right checkbox) of the property. Now click the plus button to add a child property (just like you did for grouped properties). You’ll notice that the created child property doesn’t have an extraction method, instead the parent property has. The reason is that subtracted elements does not return any value and cannot be used for rendering. The parent property is the one we should be using and the subtracted elements will be dropped from its html or text values.

Choosing the right extraction method

The extraction method is the output that will be produced from the selected element and eventually is the value of the property at run-time. Equally, array properties will have a list of values but all of them will also be produced using the selected extraction method for each selected element in the list.

The available extraction methods are:

- Text - used to extract **plain text** out of the element. This might also include text that is not visible on the original webpage or missing spaces between words and lines due to the loss of the layout and styles that are only available with rich text such as HTML. This is the default and recommended method for most cases when you just need to extract the text value of the element in order to style it by yourself using rendering templates.
- Inner HTML (i-html) - used to extract the HTML content of an element, excluding the tag and attributes of the selected element. Content inside the element will be completely extracted into HTML. Be aware that styles defined on external CSS files are not going to be extracted. Inline styles and the complete hierarchical tree of the HTML are included. This method is recommended for extracting big chunk of rich text such as article body. Once the HTML was extracted you may use your own style rules to give it a nice look.
- Outer HTML (o-html) - same as Inner HTML, but the tag and attributes of the selected element are included this time. Recommended when your selector may return multiple types of elements with different tags.
- SRC - used to extract an image source from <img> elements. You can then decorate the source value with your own image tag and styles.

- HREF - used to extract a hyperlink URL from <a> elements. You can then decorate the URL with your own hyperlink tag.
- Title – used to extract the title attribute (tooltip) from an element.

More about selectors

When using the eyedropper tool of the Cartridge Builder to pick elements, the tooltip shows you some very useful information. Just mouse hover an element to see its hierarchical location, tag name, ID, class names, suggested selectors and the number of elements found by the current selector.

Remember, you are picking a selector not an element. The goal here is to have the most reliable selector that will select the same content for similar web pages of the same website and are supported by the cartridge you are building. When you mouse hover an element, Cartridge Builder will give you suggestions for selectors that choose different combinations of elements. The hovered element will be included in each of the suggested selectors, but not the other elements. The first displayed suggestion is the most recommended, choosing as many elements as possible while keeping it reliable and avoiding false positive results. The next suggestions (if any) will decreasingly select fewer elements than before.

In some cases, none of the suggested selectors is qualified for the required results. In this case it is recommended to create a parent property and select a container that will limit and narrow the results and is logically reasonable for the layout of the webpage. Then you can create a child property and try to select the required elements. If neither this help, you may try to enter a custom selector using the full syntax available on the jQuery website. This way you may select of filter by attributes, complex hierarchical navigation and more.

Use the UP keyboard button to enlarge the selection by going up in the hierarchical tree of the DOM and select the parent of the current element.

Use the DOWN keyboard button to go back and shrink the selection by going down in the hierarchical tree of the DOM up until the original hovered element.

Use the RIGHT keyboard button to move to the next suggested selector that includes fewer elements than before by applying a more strict rule.

Use the LEFT keyboard button to move to the previous suggested selector that includes more elements by applying a less strict rule.

Use the INSERT keyboard button to use a custom selector. This will focus the cursor on a textbox where you can type your own selector. After finish typing press Enter to try the selector and see the results. You may then click Enter again to choose the selector and go back to the sidebar, or just move the mouse to the next element in order to reset your custom selector.

You can read about jQuery selectors at http://api.jquery.com/category/selectors/

Rendering Templates

After completing the data extraction phase, you now have a working set of data that you can play with. The goal here is to render a preview for the visited webpage that will be automatically added to each new discussion on similar pages. For instance, if you are creating a cartridge for articles on CNN.com, then you'll want to design a preview with the article's title and short description. You might also want to add some images as well, according to the extracted data. Try keeping the preview short and summarized, since it will be added to all new discussions created for articles on that website.

Preview templates are written using HTML in order to make their output rich and under great level of control. In addition to the standard HTML syntax, JsRender template syntax is used to access the extracted data and decorate it with HTML. You may also use flow control operations such as if-then-else and for iterations as well.

After clicking the Generate Template button, a default template is created for the properties defined on the first step. This template will just render each of the property values one after another. Each time you modify the list of properties to extract, you may generate the template again **(which will revert your template back to the default state)** or manually add the new properties to the template using the JsRender syntax which is recommended.

A basic knowledge of HTML is recommended for this step. It is also highly recommended using an external HTML editor software with color highlighting, syntax completion and validation. You can just copy and paste the HTML from the text area on the sidebar to import and export the template.

Single values

Property values which are not an array can be rendered using two ways, encoded, which is great for plain text and is completely safe, or untouched which is required for HTML extracted content such as rich text.

To render plain text value as encoded HTML, use **{{>property_name}}**

To render HTML value untouched, use **{{:property_name}}**

The result of using encoded rendering to render HTML values is a readable HTML syntax. In other words the HTML tags will be displayed as text. This is completely safe but those tags will just be ignored by the browser instead of making the text rich.

The result of using untouched rendering to render non-HTML or un-sanitized content is dangerous. Leading to unstable results. Use this option with cautious.

To render a child property of a non-array parent property you can use **{{>parent_name.child_name}}**

List values (arrays)

Array properties are used to store list of values, if the property has children then each of them will be exist for every item in the array list. Array properties are accessed using the **{{for}}** keyword.

In case of an array without children, use:

```
{{for property_name}}
      {{>#data}}
{{/for}}
```

The **#data** keyword is used to get the value of the iterated item. The **#index** keyword is used to get the index of the iterated item.

In case of an array with child properties, use:

```
{{for property_name}}
      {{>child1_name}}
      {{>child2_name}}
{{/for}}
```

An array may have child properties which are also arrays (nested arrays):

```
{{for contacts}}
      {{>name}}
      {{>title}}
      {{for phoneNumbers}}
            {{>#data}}
      {{/for}}
{{/for}}
```

Remember to decorate your template with HTML tags, otherwise you'll just get a bunch of text without a layout. Let's decorate the previous syntax:

```
{{for contacts}}
      <div>
      <h2>{{>name}}</h2>
      <h3>{{>title}}</h3>
      Phone numbers <br/>
      {{for phoneNumbers}}
            {{>#data}} <br/>
      {{/for}}
      </div>
{{/for}}
```

If-else conditions

There are many cases you want to control the flow of the rendered content by evaluating expressions based on property values. This can easily done using the **{{if}}** and **{{else}}** statements.

```
{{if property_name > 5}}
```

```
      {{>property_name}}
{{/if}}
```

You can also use else statements with another expression:

```
{{if contact.homeAddress}}
      Home address: {{>contact.homeAddress}}
{{else contact.businessAddress}}
      Business address: {{>contact.businessAddress}}
{{else}}
      The contact has no address.
{{/if}}
```

You can evaluate (or render) the number of items in array by accessing its length property:

```
{{if contact.phoneNumbers.length > 0}}
      {{for contact.phoneNumbers}}
            {{>#data}} <br/>
      {{/for}}
{{else}}
      The contact has no phone numbers.
{{/if}}
```

A complex expression may contain multiple expressions with && (and) and || (or) keywords:

```
{{if contact.disabled || contact.deleted}}
      The contact has been removed.
{{/if}}
```

You can evaluate the #index keyword to render only the first items on an array:

```
{{for contacts}}
      {{if #index < 10}}
            {{>name}} <br/>
      {{/if}}
{{/for}}
```

Parent navigation

In some complex scenarios you might need to access properties related to the current context which is not direct child properties. For instance, let's print a list of articles:

```
{{for articles}}
      {{>title}}
      {{>description}}
      article {{#index}} out of {{>#parent.data.length}}
{{/for}}
```

The #parent.data expression in this case gave us the context of the array.

In the following example we're comparing the value of the category property of each article to the root category property of the page and highlight in bold if they are the same.

```
{{for articles}}
      {{if category == #parent.parent.data.category}}
            <b>{{>title}}</b>
      {{else}}
            {{>title}}
      {{/if}}
{{/for}}
```

The #parent.parent.data expression in this case gave us the context of the root properties dictionary. The first #parent expression led us to the array property while the second one led us to the root.

Page metadata and OpenGraph

In addition to the properties you have defined on the data extraction phase, some metadata is automatically added to the basket. The meta property has the following child properties:

- meta.url - the URL of the current document as returned by the browser
- meta.title - the title of the current document as returned by the browser
- meta.opengraph - optional OpenGraph meta tags
    - meta.opengraph.site_name
    - meta.opengraph.url
    - meta.opengraph.title
    - meta.opengraph.image
    - meta.opengraph.type

Deployment

Upload cartridges

Once you've done editing your cartridge and you are happy with the preview result you are ready to upload it to the Jive community server which hosts your instance. Jive server is where the cartridge is saved, if you do not upload the cartridge you will lose all your progress when you leave the web page or close the browser tab.

Click Upload Cartridge and a dialogue will open. Type a name for your cartridge, this name will be displayed at the top of the sidebar when the cartridge is active.

The second, and the most important field is the URL pattern for activation. Here you need to type a substring of a URL that will be used to determine if a webpage is qualified for using your cartridge.

For instance, we're building a cartridge for articles on IGN, and the currently loaded webpage is:

http://www.ign.com/articles/2014/03/10/guardians-of-the-galaxy-james-gunn-chris-pratt-and-kevin-feige-reveal-new-details

We want to make sure that every other article on the IGN website will activate our cartridge, so as it logically seems the pattern should be:

http://www.ign.com/articles/

This will cause every URL that contains the above substring to evaluate successfully and activate the cartridge.

After filling both fields click Upload. The cartridge will be available to your user account only, on the selected default community. An administrator may choose to publish and share the cartridge for other users in the community.

After the upload was completed you need to refresh the webpage in order to test your cartridge.

Edit cartridges

To edit a cartridge, visit any webpage which activates the cartridge you want to edit and make sure the sidebar displays the message "{cartridge} is activated". Then click the gear icon on the sidebar and choose Cartridge Builder from the popup menu. This will open the Cartridge Builder dialogue and the data from the cartridge will be loaded. After modifying the cartridge just follow the upload process again. This will override the old version of the cartridge.

Note: You can only edit cartridges created using the Cartridge Builder and has not been published yet.

Regular Expressions

In some cases a more complex URL pattern is required to determine if a webpage is qualified for using your cartridge. When you have to check query parameters inside the URL or allow multiple patterns. This can be done using Regular Expressions which is a language for evaluating and processing strings. In order to toggle the use of a regular expression in the URL pattern you have to **prefix the expression with a $ sign**.

Twelve characters have special meanings in regular expressions and they must be encoded with a preceding backslash in order to be used as strings. The characters are \ ^ $. | ? * + ( ) [{

Let's take a few examples:

```
$^https:\/\/www\.google\.com\/$
```

Will successfully evaluate for the exact https://www.google.com/ URL. The ^ sign represents the mandatory beginning of a string and the $ sign represents the mandatory ending of a string. The address itself is encoded since / and . characters are part of the regular expression syntax so a backslash must precede each of them.

```
$www\.ign.com\/articles\/[0-9]{4}\/[0-9]{2}\/[0-9]{2}\/
```

Will successfully evaluate for http://www.ign.com/articles/2014/03/10/anytext or a similar URL width another date.

```
$www\.ign.com\/[articles|videos|reviews].\/
```

Will successfully evaluate for http://www.ign.com/articles/anytext or http://www.ign.com/videos/anytext or http://www.ign.com/reviews/anytext

More info and examples of regular expressions can be found at http://www.regular-expressions.info/quickstart.html

Manage cartridges (user)

You can view the list of all available cartridges for your default community by clicking the gear icon on the sidebar and choosing Settings from the popup menu. On the Settings page click the Cartridges tab.

The first table displays all shared cartridges that has been published by the community administrator and are available to all community members. The second table displays all private cartridges that you have uploaded and were not published by the administrator. These cartridges can be edited by going to a webpage that activates them and running the Cartridge Builder from there. They can also be deleted by clicking the delete link on the Settings page.

Manage cartridges (admin)

An administrator may choose to publish your cartridges and make them available to all users of the community.

When logged in as administrator go to the Admin Console of the Jive community, click System > Settings > Jive Anywhere and select the Cartridges tab. The table displays all the cartridges uploaded to the community. Each row has its published state. A non-published cartridge is only available to the user it was uploaded by (the owner). A published cartridge is available to all users of the community.

To Edit, Delete, Publish or Download a cartridge, click the dropdown menu at the row of the cartridge and choose the action you want. You can also upload a new cartridge by clicking the Upload new cartridge button.