Drag and Drop Dynamic Shapes Challenge: Shapey.js  
Daniel Llewellyn (Kenji776)

**Description**

Shapey.js is a jQuery plugin for creating draggable, droppable, highly dynamic shapes from a JSON data feed. It is highly flexible and can be configured to fit many different needs. By using a light weight JSON data structure it is easy to dynamically create any number of shapes with any properties and buttons. You can control css properties on shapes and buttons and the framework allows you to easily extend functionality by providing callbacks after key events.

**Usage**

Using Shapey is easy. First, include jQuery and jQuery UI in your page, then include the Shapey.js plugin. Next you need some JSON shape data. A detailed explanation of the Shapey Shape structure is provided below. Shapey also comes with a demo function you can use to get started. In your script statement after loading the Shapey library call  
  
This will print a JSON string to your javascript console (provided you are using Chrome or Firefox, or a newer version of Internet Explorer). Copy and paste the generated content into a text file. Name the file shapes.txt and put it in the same folder as your html page.

Next we need to invoke Shapey. On your html page, create a container div, where Shapey will write it’s output to. For example in your body tag include  


Then in a script tag include the following.  
  


Now Shapey is ready to rock. Put your code into a web server (it won’t run locally due to ajax security restrictions on the local machine. A lightweight web server, mongoose.exe is included in this package. Just start it in the same folder where your code is, and in your web browser go to 127.0.0.1:8080) and run the page. You should see your container div get populated with draggable, droppable shapes with some buttons.

You can of course experiment and modify the JSON data to see what happens. Try adjusting the width and height of shapes, or setting a different background image. You can also modify the various properties of the plugin such as turning destroyOnMerge on, or turning off the animations for a faster interface.

**Advanced Usage**

1. **CSS Properties on Shapes**

Now that you have Shapey running you are most likely going to want to start customizing. Some of the customizations are pretty intuitive, changing button names, sizes, etc. But what about more fine tuned control? Shapey gives you complete access to the CSS styling of the generated shapes and buttons by use of the properties key of the shape JSON data. Let’s examine a single Shapey shape.  
  


As you can see it has several properties. A name, id, type label offsets and a deleted flag. But it also has a sub object called properties. The properties key of a Shapey shape contains all the CSS rules to be applied to the shape. In this case we set a background image of Shape1.png, a width of 122 pixels, a height of 122px, and we tell the background image to scale to 100% of the div size (this is used for the grow and shrink animations that are applied when a shape is hovered over). As you can probably guess, you can list as many CSS properties as you like here. As you can imagine, this gives you a huge amount of power over the look and feel of your shape. You can also easily position the label (which is the string specified in the name) to appear anywhere on your shape.

1. **Buttons**

Shapey also gives you the ability to create as many or as few buttons on a shape as you like. Buttons are declared as an array of objects in the ‘buttons’ property of a Shapey shape in the source JSON. At minimum every button needs to a properties object (just like the one in a Shapey shape) with values for the width, height, top offset (top) and left offset (left). It is also useful t specify a click property with the name of a function to be used as a callback. Here is an example button that might be included on a shape.



You can see that buttons is an array of elements. We only see one here, but you could have as many as you like. You can see that using just pure CSS we are going to create a 20 pixel by 20 pixel button with a solid 1 pixel border and a red background. It will sit 20 pixels from the top of its parent shape, and 20 pixels to the left as well. When it is clicked, it will call a javascript function named clickHandler (which will be passed all relevant information about the event, the button as well as it’s parent). You can see Shapey allows you great flexibility in designing any kind of buttons you like, which can perform any action you like.

1. **Callbacks**

Shapey allows you to further extend the functionality of the plugin by means of four possible callback functions.

1. The merge callback
   1. This function specified in the Shapey Shape JSON by use of the ‘mergeHandler’ key gets called when two shapes are merged together. It gets passed a reference to the event, the source object (the dragged shape), and the target object (the dropped onto shape).
2. The clone callback
   1. This function gets called when a new shape is created as the result of two other shapes being merged. It gets passed the new shape, the parent it was cloned from and the second parent shape.
3. The destroy callback
   1. This function gets called when a shape is destroyed (such as when using the destroy on merge feature). It gets passed a reference to the destroyed shape.
4. The click button callback
   1. This function can be attached to buttons on the shape. This is called whenever a button is clicked. It gets passed the event, a reference to the pressed button, and a reference to the parent shape the button resides on.

Below is a sample shape that uses all possible callbacks.



1. **Plugin Invocation Argument Description**

When invoking Shapey you have several options you may set that can change the behavior of Shapey and dramatically alter its behavior. Below is an explanation of each argument.  
  


**Other things to note**

* The type attribute on the JSON source is used to control what shapes can be merged together when typeMergeMatching is on. Even shapes with vastly different properties can be merged if they share the same type.
* You can retrieve a serialized version of the current shape configuration by calling the shapey\_serializeShapeData() function. It will return a JSON string that represents the current shape config, including all added and deleted shapes.
* All the Shapey methods are publically exposed, meaning you can access the same functionality Shapey uses to create the application. You can all them from your callbacks, or wherever you like to leverage the same powerful functionality Shapey uses itself. Spend some time exploring Shapey.js to see what is available. The code is heavily commented to make it easier for you to modify it however you need.
* While Shapey offers you the maximum amount of flexibility possible, this also gives you the user abilities to cause Shapey to act oddly. If your provided CSS styles are causing Shapey to freak out, try removing properties one by one until it starts behaving normally again.
* You can directly read Shapey shape properties by accessing the global shapeyShapes object variable.
* The shapey\_shapeFactory() function could be removed in production after you have a valid JSON feed to create shapes from. It is only a helper function to get you up and rolling with Shapey by providing sample data.
* Shapey does it’s best to take care of missing data situations, but be nice to it, and provide as many of the crucial properties (id, name, width, height, etc) as you can in your JSON. This makes Shapey happy, and less likely to break in some weird way.

So there you have it. A simple, flexible plugin for creating draggable, droppable, shapes with infinite buttons and totally customizable styling. Hope you enjoy using it half as much as I enjoyed writing it.