

Adobe XD Artboard Plugin

How to insert shapes on the Artboard:

This documentation explains step by step instructions on how to create a plugin dialog box to add personalized shapes onto an artboard in Adobe XD.

Prerequisites:

Basic knowledge of JavaScript and HTML

[Developer Console Debugging](#)

A text editor to write your code in (VSCode, Atom, Text, Sublime, etc)

[Set up tutorial](#)

Development Steps:

Note: Complete code for this plugin can be referenced on [GitHub](#)

1. Create your plugin scaffold

First, setup the Adobe XD environment using the setup tutorial referenced above.

Next, create a `manifest.json` file inside the developer file in the chosen IDE.

Replace the `uiEntryPoints` file of the `manifest.json` with the following:

```
"uiEntryPoints": [  
  {  
    "type": "menu",  
    "label": "Insert Shapes",  
    "commandId": "shapes",  
    "shortcut": {"mac": "Cmd+Shift+T", "win": "Ctrl+Shift+T"}  
  }  
]
```

Reference to each of the entries used above can be found in the manifest documentation along with other key components that are important for the plugin to be functional.

Next, update the `main.js` file, which was created inside the developer file, and update the manifest's `commandId` to a handler function.

Add this section of code to your `main.js` file:

```
module.exports =  
{  
  commands:  
  {  
    shapes: function ()  
    {  
      dialog.showModal();  
    }  
  }  
}
```

2. Creating Elements

In JavaScript, elements are a general base class from which elements are inherited in a document. In the function defined below, `createElement` is being used to create an element node with a specified name.

Another important aspect of the function below is the use of `props` which is used to pass data from a parent component to the `children` component.

The function below is essentially a shortcut of creating and styling the elements you wish to add to the upcoming dialog box.

Add this section of code to your `main.js` file to setup the elements shortcut:

```

/**
 * Shorthand for creating Elements.
 * @param {*} tag The tag name of the element.
 * @param {*} [props] Optional props.
 * @param {*} children Child elements or strings
 */
function h(tag, props, ...children) {
  let element = document.createElement(tag);
  if (props) {
    if (props.nodeType || typeof props !== "object") {
      children.unshift(props);
    }
    else {
      for (let name in props) {
        let value = props[name];
        if (name === "style") {
          Object.assign(element.style, value);
        }
        else {
          element.setAttribute(name, value);
          element[name] = value;
        }
      }
    }
  }
}

```

3. Create Dialog Box

Since the plugin will be activated using a dialog box, a key component to start with is **let dialog** in which we will personalize the plugin.

To begin, specify a local variable using **let** which will determine the width between the label and input.

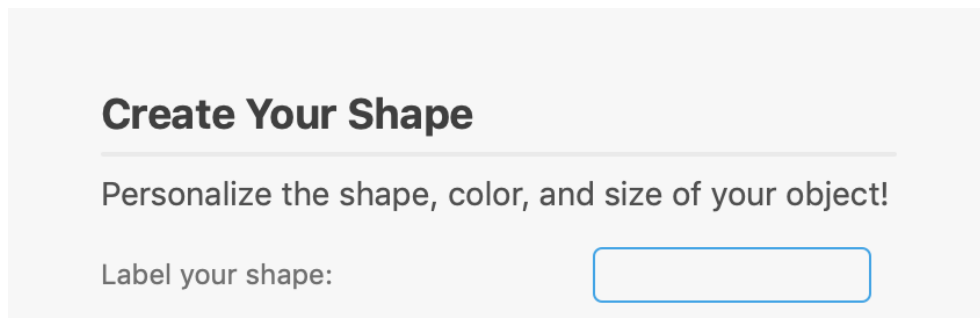
The dialog box consists of four components which are the shape label, shape type, color, and size.

Create a let dialog and insert the following sections of code for each of the components in your main.js file.

Below is the snippet of code for the header of the dialog box along with the shape label component.

```
h("form", {method: "dialog", style: {width: 355 }, onsubmit },
  h("h1", "Create Your Shape"),
  h("label", {class: "row" },
    h("p", "Personalize the shape, color, and size of your object!"),
    h("label", {class: "row" },
      h("span", { style: { width: label_width } }, "Label your shape: "),
      label_shape = h("input", { uxpQuiet: true, style:{ flex: "1" } })
    ),
  ),
```

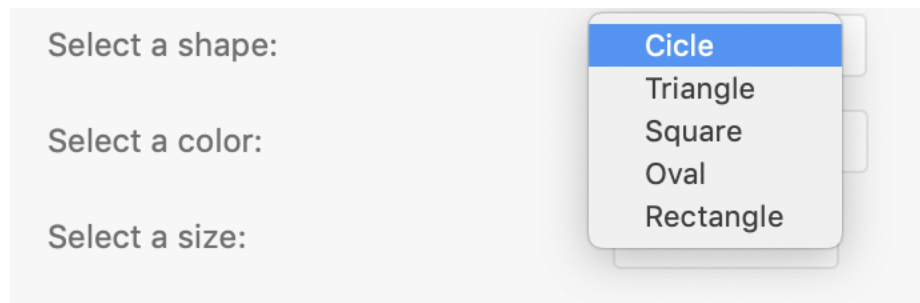
Dialog box output implementing the code above:



Below is the snippet of code for selecting the shape type:

```
h("label", { class: "row"},
  h("span", {style: {width: label_width } }, "Select a shape:"),
  shape = h("select",
    h("option", { value: "Circle"}, "Circle"),
    h("option", { value: "Triangle"}, "Triangle"),
    h("option", { value: "Square"}, "Square"),
    h("option", { value: "Oval"}, "Oval"),
    h("option", { value: "Rectangle"}, "Rectangle")
  ),
)
```

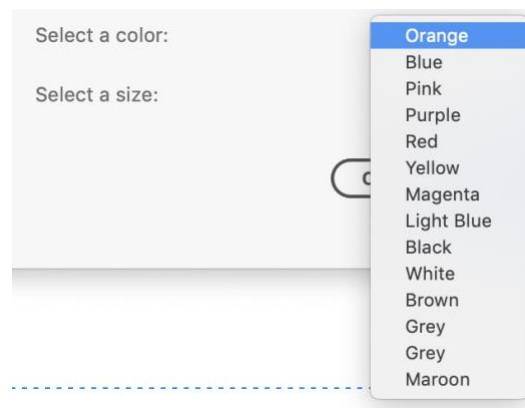
Dialog box output implementing the code above:



Below is the snippet of code for selecting the color:

```
h("label", { class: "row"},
  h("span", { style: { width: label_width } }, "Select a color:"),
  color = h("select",
    h("option", { value: "Orange"}, "Orange"),
    h("option", { value: "Blue"}, "Blue"),
    h("option", { value: "Pink"}, "Pink"),
    h("option", { value: "Purple"}, "Purple"),
    h("option", { value: "Red"}, "Red"),
    h("option", { value: "Yellow"}, "Yellow"),
    h("option", { value: "Magenta"}, "Magenta"),
    h("option", { value: "Light Blue"}, "Light Blue"),
    h("option", { value: "Black"}, "Black"),
    h("option", { value: "White"}, "White"),
    h("option", { value: "Brown"}, "Brown"),
    h("option", { value: "Grey"}, "Grey"),
    h("option", { value: "Grey"}, "Grey")
```

Dialog box output implementing the code above:

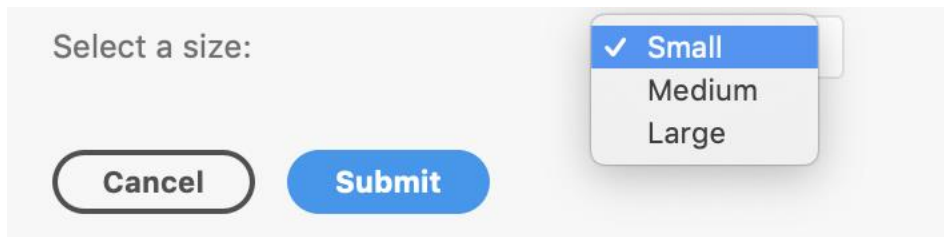


Below is the snippet of code for selecting the size of the shape and the footer:

```
h("label", { class: "row"},
  h("span", {style: {width: label_width } }, "Select a size:"),
  size = h("select",
    h("option", { value: "Small"}, "Small"),
    h("option", { value: "Medium"}, "Medium"),
    h("option", { value: "Large"}, "Large")

  )
),
h("footer",
  h("button", { uxpVariant: "primary", onclick(e) { dialog.close( ) } }, "Cancel"),
  h("button", { uxpVariant: "cta", onclick(e) { dialog.close( ) } }, "Submit")
)
```

Dialog box output implementing the code above:



On the left is the empty dialog box and to the right is the filled one:

Create Your Shape

Personalize the shape, color, and size of your object!

Label your shape:

Select a shape:

Select a color:

Select a size:

Create Your Shape

Personalize the shape, color, and size of your object!

Label your shape:

Select a shape:

Select a color:

Select a size:

Instructions:

1. Open Adobe XD
2. Click on Plugins
3. Next, click Artboard Features
4. Click Insert Shapes
5. In the dialog box, enter a label for your shape
6. Click on the scroll down arrow to select the shape, color, and size
7. Click Submit

Personal Reflection:

The idea behind this plugin is for users to be able to insert a shape of their choice directly onto their artboard instead of making it from scratch. If a designer has a specific image in mind, for example a small pink circle, then using this plugin they would be able to input their specification and have a shape appear. As of now the submit button does not implement a shape onto the artboard, however as I continue to work on this plugin, I will add in functions for each of the shape and code in the different styling options. Furthermore, I would add in a color spectrum, so users are not limited to just the options in the dropdown menu.

Building this plugin was a fairly smooth process and one that I enjoyed thoroughly. Initially, I began this project by brainstorming ideas of plugins which I would appreciate as a user. This led me to create a sketch on paper and then bring that vision to life on Adobe XD. Before I started coding, I spent my time researching and reading up on the documentation referenced on the Adobe XD

website and got insight on where to start and how to build my plugin. Whilst coding this plugin, the main issue I ran into was understanding how the `onsubmit()` function operates and how to incorporate that into the `dialog` code. I spent hours debugging using the developer console and researching the meaning of the errors. Some errors I was receiving were routing errors which were hard for me to navigate through. I had to include breakpoints in my code to narrow down to find the cause of each console error I was receiving. After going through my code line by line I noticed it was an error due to an extra variable I had used but did not declare within the `onsubmit` function. Regardless of the error being related to syntax, it gave me an opportunity to not only thoroughly understand the code but also led me to dive deeper into different functions and API's that can be used to further continue this project.