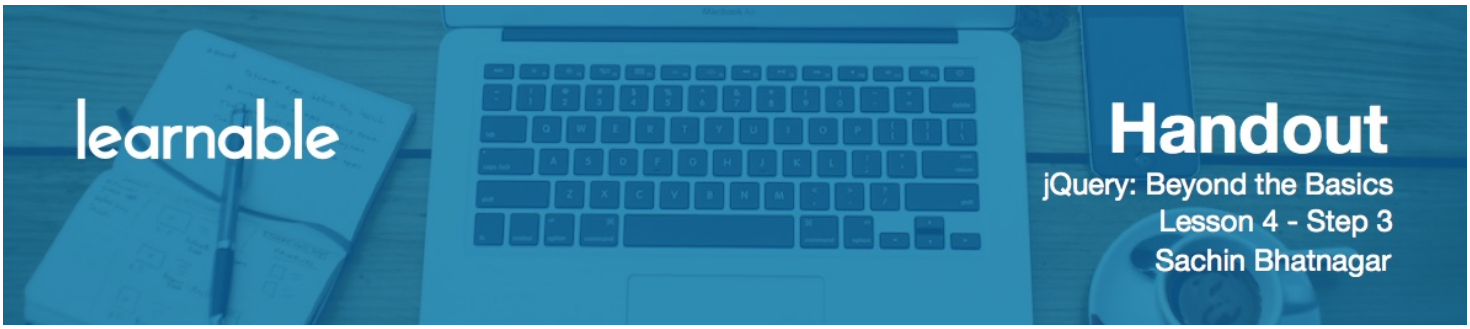

Introduction to modal dialogs

Modal dialogs are used to grab user attention. Using JavaScript, you can easily invoke the `alert` modal dialog which can be used to display a message. If you want the user to provide a feedback action to your message, you can invoke the `confirm` or the `prompt` dialogs.

However there are two major things to keep in mind:

- The look and feel of these dialogs depend on your operating system and the browser that you're using.
- These modals are blocking, which means that they stop the JavaScript event loop as long as they're visible. This effectively means that your code will no longer be able to work in the background as long as the dialog is visible, and this can get in the way of asynchronous functions.

So why not use jQuery instead to create these dialogs? For this purpose we use HubSpot's Vex modal dialog plugin.

Working with modal dialogs in Vex

Open up *Modal_Begin.htm* to proceed and github.hubspot.com/vex/docs/welcome page.

We have a simple user interface with four buttons. The first three are designed to invoke one of Vex's dialogues: a confirm dialogue box, a simple alert that just displays a message and a prompt dialog that requires the user to type in his or her name. The "Open box" button will be used to demonstrate a dialog box which can be fully customized to include multiple form fields and custom buttons.

First of all download the Vex plugin. Inside the *css* folder there is a *vex.css* file which contains all the base styles. There are also a couple of theme files that allow you to change the look and feel of the dialogs.

Within the *js* folder there are many JavaScript files, but we only need the *vex.combine.min.js* file for this project. It contains all of the features that Vex has to offer as a whole. So copy this file into our project's *js* folder; also also copy all the *.css* files into our *css* folder.

Hook up all the required files:

```
<link rel="stylesheet" href="css/vex.css">
<link rel="stylesheet" href="css/vex-theme-flat-attack.css">
<script src="js/vex.combined.min.js"></script>
```

Building a confirm dialog box:

Let's build a confirm dialog box first. Imagine you're using this confirm dialog on a ticket booking website and before the user commits to the booking you want to give them an option to go ahead with the booking or cancel:

```
$("#confirmBtn").click(function() {
  var confirmOutput = $("#confirmOutput");
// Create a VEX Confirm box here...
  vex.dialog.confirm({
    message:"Should we confirm your booking ?",
    callback: function(value) {
      if(value) {
        confirmOutput.text("Thank you. Your tickets have been booked !");
      } else {
        confirmOutput.text("Maybe next time !");
      }
    }
  });
});
```

We define a callback function which provides us with a value that is either `true` or `false`.

Building an alert box

Next let's create an alert box:

```
$("#alertBtn").click(function() {
// Create a Vex Alert box here...
  vex.dialog.alert("Hello Everyone ! This is an Alert Box !");
});
```

Building a prompt dialog box

Next the prompt dialog box which contains a single input field for the user to type in:

```
$("#promptBtn").click(function() {
// Create a Vex Prompt box here...
  var promptOutput = $("#promptOutput");
  vex.dialog.prompt({
    message:"What is your name ?",
    placeholder:"Type in your name...",
    callback:function(value) {
      if(value){
        promptOutput.text("Hello " + value + "! How are you doing ?");
      } else {
        promptOutput.text("You forgot to type in your name ! :)");
      }
    }
  });
});
```

Building a customizable dialog box

Vex allows you to create fully customizable dialogs as well:

```
$("#openBtn").click(function() {
// Create a Custom Vex Dialog Box here...
  var optionsOutput = $("#optionsOutput");
  vex.dialog.open({
    message:"<h3>Register Now</h3><hr>",
    input:"<label>Name : <input type='text' name='username' placeholder='Your Name' required></label>\n<label>Date of Birth : <input
type='date' name='dob' placeholder='Date of Birth' required></label>\n<label>E-Mail : <input type='email' name='email' placeholder='E-
Mail' required></label>",
    buttons:[
      $.extend({}, vex.dialog.buttons.YES, { text:'Register' }),
      $.extend({}, vex.dialog.buttons.NO, { text:'Cancel' })
    ],
    callback:function(value) {
      if(value){
        optionsOutput.text("Hello " + value.username + "! Your E-mail ID is " + value.email + " and your date of birth is " +
value.dob);
      }
    }
  });
});
```

`input` is a property used to define the custom input fields that we want to place in our dialog box. For this example we have three fields here for the name, date of birth, and the user's email address.

We also change the text on the default buttons that we will get on the dialog box. This is done by the **buttons** property. The `$.extend` method is used to merge properties of two objects, so we first pass in an empty object, which essentially means that we want to preserve the objects, and hence we are not defining a target object to which to set the merged properties to. Next, we take the `vex.dialog.buttons.YES` object, and merge it with another object where we are setting. The same is done for the `vex.dialog.buttons.NO` object.