Popup Modal Box – Design from Scratch

Popup modal boxes are quite popular among almost all web developers these days. In simple terms, it is an overlay of HTML element(s) over a web page. If you an experienced web developer, it is not a new thing for you. You may already familiar using modal boxes that come with many third-party libraries such as jQuery.

But, have you ever wondered whether you can design a custom modal box for your projects? Are you curious about what is happening behind modal boxes? If YES and you are not sure how; this tutorial is for you. If you have doubts, find out how easy it is by going through this. In this tutorial, I'll take you in the process of creating a custom modal box using only HTML, CSS and some JavaScript without the help of any third-party libraries. Then, I'll show you how to alter your design to come up with different types of modal boxes.

Do We Need Modal Boxes?

Usually, a modal box is designed to direct the users' attention to a certain detail from the web page without redirecting the user to another page or reloading the page. When it is required to highlight something to the user, the modal box comes to play.

Showing messages related to a transaction just made by the user, requesting confirmation to a specific task that the user going to perform, prompt user for some data to continue a transaction, inform

the user that the page is still loading its content, show web forms such as login/ signup, enlarge images in the gallery, play videos and display pop-up advertisements are some of the most common usages of modal boxes. Now you can decide whether you need a modal box to your website.

Can We Create a Modal Box from Scratch?

YES, if you know little about HTML, CSS and JavaScript, it’s not a big deal. You heard it right. You don’t need to be a web developing mastermind to make a modal box from scratch without using any libraries. I am sure that, once you know how it is done, you will always love to create your own modal box.
First of all, we have to decide on the layout of our modal box. We have to think of what content we are going to show in our modal box and how it should look like with them. Based on the need and the content, it can be categorized into several different types. Some of them are,

- Modal dialog boxes
- Confirm boxes
- Prompt boxes
- Web forms
- Image/ Video preview boxes and
- Advertisements.

So, let’s get on the play with some coding.

Design the Modal Box Layout

Let’s focus on modal dialog box first. So, let’s create a structure to have a heading (title), a message, a close icon and an “OK” button. The HTML markup used in this example is as follows:

```
<div id='modal_box' class='modal_box'>
    <div id='modal_box_close' class='modal_box_close'>&#9
587;</div>
    <div id='modal_box_title' class='modal_box_title'>Mod
al Title</div>
    <div id='modal_box_message' class='modal_box_message'
>
        I am a simple modal box with HTML + CSS + JavaScr
ipt.
    </div>
    <div id='modal_box_footer' class='modal_box_footer'>
        <div class='button'>OK</div>
    </div>
</div>
```

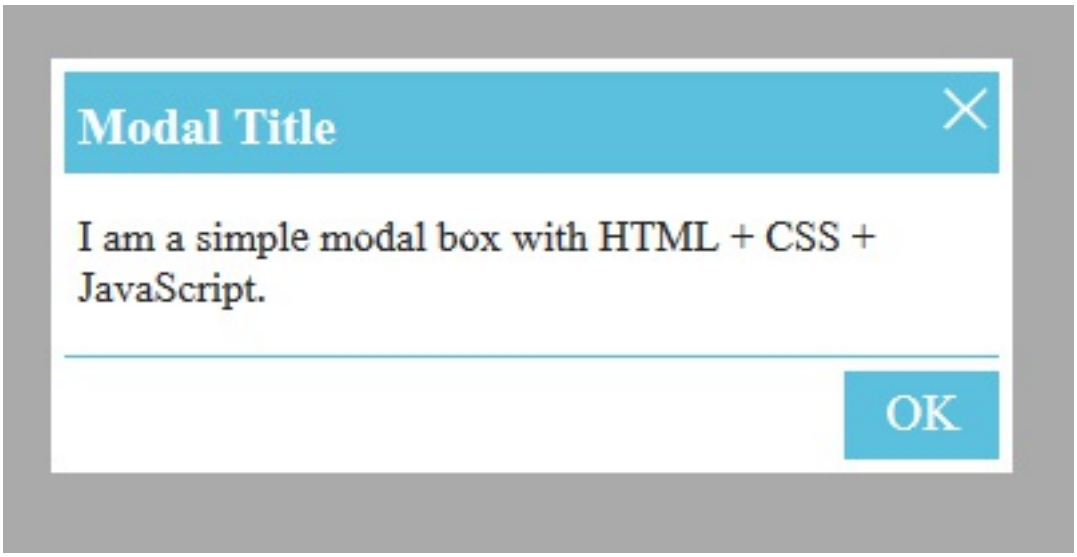
Now we have the structure. But, it’s not so pretty at the moment. So, let’s add some styles:

```
.modal_box {
  width: 30%;
  position: fixed;
```

```
  top: 50px;
  left: 35%;
  display: none;
  border: none !important;
  z-index: 10;
  padding: 5px;
  background: #FFFFFF;
}
.modal_box .modal_box_close {
  float: right;
  padding: 0 5px;
  cursor: pointer;
  color: #FFFFFF;
  font-weight: bold;
}
.modal_box .modal_box_close:hover {
  color: #232323;
}
.modal_box .modal_box_title {
  padding: 10px 0 5px 5px;
  font-size: 1.2em;
  font-weight: bold;
  color: #FFFFFF;
  background: #5BC0DE;
}
.modal_box .modal_box_message {
  padding: 15px 5px;
  color: #232323;
```

```
  background: #FFFFFF;
  border-bottom: 1px solid #5BC0DE;
}
.modal_box .modal_box_footer {
  padding: 5px 0 0;
  text-align: right;
}
.modal_box .modal_box_footer .button {
  display: inline-block;
  padding: 5px 15px;
  text-align: center;
  cursor: pointer;
  font-size: 1.2em;
  background: #5BC0DE;
  color: #FFFFFF;
}
.modal_box .modal_box_footer .button:hover {
  background: #3BB8DF;
  color: #FFFFFF;
}
```

Now our modal dialog looks like this:



Let’s Add Some JavaScript

Our next step is to add some functionality to show/ hide our modal dialog. Here I am going to show you do it with basic JavaScript by adding two functions called **‘showModal’** and **‘hideModal’**.

```
// Modal Box
var modalBox = document.getElementById('modal_box');

// Show our modal box
function showModal(){
  modalBox.style.display = 'block';
}

// Hide our modal box
function hideModal(){
  modalBox.style.display = 'none';
}
```

Let’s add a button to check the functionality of our modal and update our markup by adding click event handlers to close icon and ‘OK’ button.

```
<button onclick='showModal()'>Show Modal</button>

<div id='modal_box_close' class='modal_box_close' onclick
='hideModal()'>&#9587;</div>

<div class='button' onclick='hideModal()'>OK</div>
```

The modal box is working. But, didn’t you notice that something is not right yet? Hmm, we still have access to the web page even when the modal box is opened. So, how are we going to fix that? It’s really easy. We just need to add an overlay to cover the whole web page. Following is the markup and style for that:

```
<div id='modal_wrapper' class='modal_wrapper'></div>
```

```
.modal_wrapper {
  display: none;
  opacity: 0.7;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
```

```
  z-index: 10;
  background: #333333;
}
```

Next, we update our JavaScript to show/ hide overlay with the modal box.

```
// Overlay wrapper
var modalBoxWrapper =  document.getElementById('modal_wra
pper');
// Modal Box
var modalBox = document.getElementById('modal_box');

// Show our modal box
function showModal(){
  modalBox.style.display = 'block';
  modalBoxWrapper.style.display = 'block';
}
// Hide our modal box
function hideModal(){
  modalBox.style.display = 'none';
  modalBoxWrapper.style.display = 'none';
}
```

This is the complete example for our simple modal dialog box up to now.

https://codepen.io/dumindalk/pen/WoWGNJ

Let’s improve our modal dialog to change the title and message dynamically. We have to modify the **‘showModal’** function as follows:

```
// Show our modal box with title and message
function showModal(args){
// Show our modal box with title and message
function showModal(args){
    // Set title of the modal box
    title = (typeof args !== 'undefined' && typeof args.t
itle !== 'undefined')?args.title:'Modal Title';
    var modalTitle = document.getElementById('modal_box_t
itle');
    modalTitle.innerHTML = title;

    // Set Message of the modal box
    message = (typeof args !== 'undefined' && typeof args
.message !== 'undefined')?args.message:'I am a modal dial
og with HTML + CSS + JavaScript';
    var modalMessage = document.getElementById('modal_box
_message');
    modalMessage.innerHTML = message;

    modalBox.style.display = 'block';
    modalBoxWrapper.style.display = 'block';
}
```

Now we can call our **‘showModal’** method in 3 ways.

1. *showModal()* – without any parameters. It'll load the modal dialog with the default title and the default message.
2. *showModal({**title:'title'**})* – with only title parameter. It'll load the modal dialog with the given title and the default message.
3. *showModal({**title:'title', message:'message'**})* – with both title and message parameters. It'll load the modal dialog with given title and message.

Below is the working example:
https://codepen.io/dumindalk/pen/QGPRQw

Enhance the Modal Box

Add Themes to Dialog Box

Now we have a decent modal dialog which is capable of changing its title and message based on the given data. How about if we add some themes to the modal dialog? Isn't that interesting? So, let's change styles our modal dialog to show different types of messages (Information, Warning, Success and Error) without changing the layout.

We will achieve this by adding a class name to our modal box markup dynamically using JavaScript. Before that, we have to define new rules for those classes in our CSS. I have used *"info"*, *"warning"*, *"success"* and *"error"* classes in this example. Default style would be "info". I have to change the background color of the title, OK button and the border color to change the themes.

To update the class name of modal box, the following JavaScript is

used:

```
// Set theme of the modal box. [default is info]
theme = (typeof args !== 'undefined' && typeof args.theme
 !== 'undefined')?args.theme:'info';
modalBox.className = 'modal_box '+ theme;
```

To show the modal box with different themes, now you have to add “theme” entry to the argument list of the showModal method. So, now the call to showModal would be as follows (for a success type message).

```
showModal({title:"New Title", message:"I am modal box and
 this is a success message", theme:"success" });
```

You can see the working sample here:
https://codepen.io/dumindalk/pen/WoWBVr

Change Display Position and Size

Our modal box is displayed 50px from top and 35% from left. Also, its width is set to 30% to center align horizontally. Sometimes, it may be important to change the position where we display the modal box and change the size of it based on the usage. There are two ways to do that. One is by changing the width, top and left values of the CSS. The other one is changing the position dynamically by JavaScript.
To change the position and size with CSS, you can simply change the

following values.

```
.modal_box{
width: 30%;
    top: 50px;
    left: 35%;
}
```

To change the position of the modal box with JavaScript, you can update the top, right, bottom and left properties of DOM style object inside the showModal function. An example usage would be like;

```
modalBox.style.top = ‘20%’;
modalBox.style.left = '0';
```

To change the size of it, you can simply update the width property of DOM style like below.

```
modalBox.style.width = '45%';
```

Add Callbacks

Our modal dialog is good looking now. However, it is only capable of display some information to the user and closes when the user clicks on the close icon or OK button. But, in some situations, we want to perform some task after the user clicks on the OK button. That’s where

we need to use callbacks. Let’s improve our modal box to handle a callback function.

To begin with, let’s remove the ‘onClick’ attribute from the OK button and add an ID to it. We will use this ID in our JavaScript to identify the OK button. So, our markup of the OK button is as follows:

```
<div id='modal_box_ok_button' class='ok_button button'>OK
</div>
```

Then we modify our ‘showModal’ method to get another parameter. It’ll look like below:

```
function showModal(args, ok_callback){
// code goes here
}
```

Now we can add our click event listener to the OK button. I am going to define references to OK buttons and function to be used in the callback in the global scope.

```
// OK button
var okButton = document.getElementById('modal_box_ok_butt
on');
// Function to be used in the callback.
// This is important to remove the listener in hideModal
var _ok = function(){};
```

Then we can update the *showModal()* method with the following code:

```
// Add the OK callback
okButton.addEventListener('click', _ok = function(){
    if(typeof ok_callback === 'function'){
        ok_callback();
    }
    hideModal();
});
```

Below is an example usage of our showModal function to support the callback function.

```
showModal({title:"New Title", message:"message"}, callbac
k);
```

You could either use a function reference as the callback or an anonymous function like below.

```
showModal({title:"New Title", message:"message"}, functio
n(){// do something here.});
```

It is important to remove the click event listener of OK button in *hideModal*. As the user can close the modal box using either the close icon or OK button, previously bounded listener would remain unless we remove it from there. So, the updated *hideModal* function would be

as follows.

```
// Hide our modal box
function hideModal(){
    // Remove the click event listener of OK button (if a
ny)
    okButton.removeEventListener('click', _ok);
    modalBox.style.display = 'none';
    modalBoxWrapper.style.display = 'none';
}
```

Working example with the use of callbacks is below.

https://codepen.io/dumindalk/pen/QGRLLP

There are few important things to note down when adding and removing event listeners. However, I am not going to discuss them here as it is not our topic. But, if you are interested, you can find more information about that from here.

Add Event Listener

Remove Event Listener

> ***Note:*** *Another important thing to remember here is that our custom modal is* ***NOT A REPLICA*** *to* ***default browser alert or confirm****.*

Default alert or confirm *will halt the execution* of the remaining script where we cannot achieve that behavior with our custom modal. That’s where the callback functions are really important to perform the tasks

that should execute after user interacts with the modal.

Now you know how to create a modal box, change its content, change themes and add callback functions to execute after the user clicks a button. But, we made a modal dialog box which is only capable of display messages to the user. What about the other types of modal boxes? Can we make them like this easy? Yes, of course. We can make them too. I'll show you how to design confirm modal, prompt modal and HTML content. (such as web forms).

Create Confirm Box

When it comes to **confirm dialog box**, basically user has two options, **YES** or **NO**. Simply, we have to add another button and a handler function to our modal box to make it a confirm dialog. In our example confirm box, I'll use separate markup and JavaScript functions to show and hide it (*showConfirm*/ *hideConfirm*) with better clarification.

```
<div id='confirm' class='modal_box'>
    <div class='modal_box_close' onclick='hideConfirm()'>
&amp;#9587;</div>
    <div id='confirm_title' class='modal_box_title'></div
>
    <div id='confirm_message' class='modal_box_message'><
/div>
    <div id='confirm_footer' class='modal_box_footer'>
        <div id='confirm_yes_button' class='button yes_bu
```

```
tton'>YES</div>
        <div id='confirm_no_button' class='button no_butt
on'>NO</div>
    </div>
</div>
```

> ***Note:*** *I have changed the IDs of this markup. But, use the same CSS classes for styles.*

You may notice that the only major difference of this markup structure is that there is another button in the footer. Following CSS used to change the look and feel of the new buttons:

```
. yes_button {
  background: #5CB85C !important;
  color: #FFFFFF !important;
}
. yes_button:hover {
  background: #3EBA3E !important;
  color: #FFFFFF !important;
}
.no_button {
  background: #D9534F !important;
  color: #FFFFFF !important;
}
.no_button:hover {
  background: #D73933 !important;
```

```
    color: #FFFFFF !important;
}
```

You can see a working example below.

https://codepen.io/dumindalk/pen/eBabex

When considering about the JavaScript, I have updated the *showConfirm* function to accept two callback parameters for YES and NO respectively.

Create Prompt Box

Next, we are going to make a prompt modal box. Addition to the confirm box, it'll contain an input field where the user will be able to submit some data. The button text would be "Submit" and "Cancel" instead of "Yes" and "No". When the user clicks on submit button, the value of the input field should be captured and returned. It should be available to use in submit callback. If the user clicks on the cancel, it'll close the modal without further process.

We can add the input field either to the markup of the modal box or dynamically using JavaScript. In this example, I'll append the input field using JavaScript. Also, I have updated the CSS to give some styles to our new input field.

Checkout a working sample here:

https://codepen.io/dumindalk/pen/NbZRww

Load a Web Form in Modal Box

Another popular item used inside modal boxes is web forms. In this example, I'll show you how to display a web form already exists in current HTML markup inside a modal box. For this, I used the showForm JavaScript function which takes the ID of the HTML form that needs to display inside the modal box.
The process of showing the web form that is already there in the HTML markup is very simple. Here, I generate the HTML element to be used as the modal box using the following code and give it ID and CSS class names.

```
var formModal = document.createElement('div');
formModal.id = 'form_modal';
formModal.className = 'modal_box';
```

Then, fill the close icon using the code below.

```
formModal.innerHTML = "<div class='modal_box_close' oncli
ck='hideForm()'>&#9587;</div>";
```

After that, I have cloned the web form in the markup and append it to the formModal using the following code.

```
var form = document.getElementById('login_form');
formModal.appendChild(form.cloneNode('true'));
```

Then, hide the original web form in the markup and show our new

modal box with the web form done using the following code.

```
form.style.display = 'none';
document.body.appendChild(formModal);
formModal.style.display = 'block';
modalBoxWrapper.style.display = 'block';
```

You can see a working example below.

http://codepen.io/dumindalk/pen/mOZRea

Even we used a web form in this example; we are able to use any kind of HTML content in the modal box using the same function. You may use it for preview images/ videos, banner advertisements, signup forms or any markup you wish to have on your website.

How to Improve Our Modal Box

Now we know creating several types of modal boxes. But, you may wonder whether it is suitable to use in a live project. Yes, we need to improve our code to make it better to use on a live website. For learning purposes, we have used different HTML markup structures and JavaScript functions to show/ hide modal boxes.

If we generate the elements within the JavaScript, we can skip the step of adding any HTML markup to the page we are going to create the modal box.

Another improvement is to bundle our JavaScript functions to a tiny library make it more professional. You can reuse this library independently whenever required. I am not going to discuss

implementing JavaScript libraries here as it would be another lengthy topic.

Conclusion

Now you have a better idea of the workaround of making a modal box on your own. Next time you need one, you can do a better benchmarking whether to use one from external lib or use your custom one.

So, get on the play with some front-end stuff. *Happy coding!*