Mastering jQuery UI

With browsers becoming more and more powerful each day, the jQuery UI library provides tools that can be used to create feature rich and interactive applications without bearing the pain of writing numerous lines of JavaScript.

Mastering jQuery UI will help you explore the depths of the jQuery UI library. Right from the first chapter, you will build your own projects while learning the intricacies of the library. Each chapter explores a new topic and covers some related components. By gaining confidence from the initial chapters you will proceed ahead to create complex applications like the Tabbed News reader utilizing the AJAX and Reddit APIs. As a bonus, you will also learn about the best and most proven practices earned through years of diligent work on the jQuery UI platform.

Each chapter is an independent mini project in itself covering different components from jQuery UI. You won’t have to wait long to actually build something functional.

Who this book is written for

If you are a frontend developer with considerable knowledge of jQuery UI and want to take this expertise to the next level, then this book is for you.

What you will learn from this book

- Create mashups using APIs of popular websites such as Reddit and Google Maps
- Integrate AJAX with different components
- Create projects that solve real-world problems and are not theoretical
- Discover the best usage of components in a given situation
- Use effects to make the UI more interactive
- Use JSONP to make cross-origin AJAX requests
- Get to grips with best practices to use while developing with jQuery UI

Become an expert in creating real-world Rich Internet Applications using the varied components of jQuery UI

Mastering jQuery UI

Vijay Joshi


$ 49.99 US
£ 36.89 UK

Prices do not include local sales tax or VAT where applicable

Open Source

PUBLISHING

Perfect Bound Cover Template

Nature of Data: Graphic Design

Document Size: 19" x 12.5"

350 x 453mm

7.50 x 9.25

235 mm x 191 mm

653

16.68mm

7.50 x 9.25

235 mm x 191 mm

Content Type: Black & White

Paper Type: White

Page Count: 312

File Type: InDesign

Request ID: CSS512458882
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 "Designing a Simple Quiz Application"
- A synopsis of the book’s content
- More information on Mastering jQuery UI

About the Author

Vijay Joshi is a web developer who lives in New Delhi, India, with his wife and daughter. Coming from a small town named Pithoragarh, Uttarakhand, in the Himalayas, he fell in love with coding while in college. He has also loved PHP and JavaScript/jQuery since his early coding days. Vijay believes that if you are passionate and enjoy your work, it becomes more of a hobby that is not boring and it never feels like a job. After freelancing for a few years, he founded a web development firm called Developers Lab along with two of his friends in New Delhi, where they build products for the travel industry and create custom web applications. Vijay is also the author of PHP jQuery Cookbook, Packt Publishing, and the technical reviewer of PHP AJAX Cookbook and jQuery UI 1.8: The User Interface Library for jQuery. When not coding, he likes to read, spend time with his family, blog occasionally at http://vijayjoshi.org and dream about getting back in shape.

Writing a book is a long and complicated task and it requires the support and coordination of many people. I am thankful to the entire team at Packt, especially my content development editor, Athira Laji, for being so cooperative and patient with me.

A big thank you to all the technical reviewers as well, who helped me immensely in increasing the overall quality of this book.

Lastly, I am feeling both proud and excited to be able to contribute to the open source community that made me what I am today.
Mastering jQuery UI

jQuery UI needs no introduction for itself. As the official website says, "jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library." It is very likely that any developer who has worked with jQuery would have used at least one of the jQuery UI widgets.

Since it is a large library having many components with each component having multiple options, it is common for developers to get confused. Despite having excellent documentation, there are only a few resources that describe jQuery UI using its practical applications. To become an expert, you must know how jQuery UI fits with real world applications and which of its components have to be used when a use case is provided.

This is the goal of this book. Our aim is to improve your knowledge of jQuery UI to a master level, so that you can use it in complex real world projects with ease.

Each chapter of the book is a mini project in itself. There are detailed step-by-step instructions along with helpful pictures that will guide you through each chapter. The chapters are not arranged in any particular order, so you may pick up any one at random.

I am optimistic that this book will help you take jQuery UI skills to the next level.

Happy Coding!

What This Book Covers

Chapter 1, Designing a Simple Quiz Application, makes use of jQuery UI's interaction components to create a quiz application. You will learn about jQuery UI's sortable, draggable, and droppable components in this chapter.

Chapter 2, Building a Jigsaw Puzzle Game, teaches you to create a Jigsaw puzzle game using some of the jQuery UI's interaction components. You will learn to divide an image into multiple tiles along with sortable techniques.

Chapter 3, Creating a Website Tour, will create a user-friendly tool to tour different parts of a web page. You will learn about jQuery UI's tooltip component and some other components in this chapter.

Chapter 4, Creating a Tabbed News Reader, teaches you to create a news reader using an external API. Using the reddit API, you will learn about creating a mashup with jQuery UI's tabs and dialog components.

Chapter 5, Implementing CAPTCHA using Draggable and Droppable, deals with creating CAPTCHAs. Using jQuery UI's draggable and droppable components, you will be able to create three different CAPTCHA implementations.
Chapter 6, *Creating an Event Timeline Using a Slider*, will teach you to create an interactive timeline to visualize events that occurred in different years. You will also learn different techniques of using the slider component.

Chapter 7, *Using jQuery UI with Google Maps API*, teaches you to use jQuery UI components with Google Maps. You will learn to control markers on a map using a slider. You will also learn to control the zoom level using a spinner.

Chapter 8, *Creating a Photo Album Manager*, explains how to create a photo album to display pictures. Users will be able to edit the names of any picture as well as delete and/or rearrange pictures in an album.

Chapter 9, *Creating Widgets Using the Widget Factory*, teaches you to create new widgets. In detailed steps, you will learn to create two different widgets to cover all aspects of the jQuery UI widget factory.

Chapter 10, *Building a Color Picker with Hex RGB Conversion*, creates a simple color selector tool. Along with RGB to Hex conversion, this chapter will guide you in creating a tool that will allow you to change the text as well as the background color of a page, using multiple slider and spinner components.

Chapter 11, *Creating a Fully Functional Dashboard*, puts together all the learning acquired through the previous 10 chapters to create a dashboard with multiple widgets. You will learn to create six different widgets in this chapter.

Appendix, *Best Practices for Developing with jQuery UI*, guides you with the best possible ways to create rich applications. This chapter will also teach you about the best coding practices and optimizations.
Designing a Simple Quiz Application

To begin with, we will design a simple quiz application where users have to match country names to their corresponding capital names by dragging a country name over its correct capital. A correct match will result in one point, and drag and drop will be disabled for both the country and its capital.

The display will contain two columns, the first one will have country names and the second one will have the names of the capitals of those countries. Both the columns will be shuffled randomly. Once the user has matched all the names successfully, a modal dialog will appear. Users will also be given the option to reset the quiz. Resetting will restart the game and shuffle the lists again.

Setting up jQuery UI

We will need the jQuery and jQuery UI libraries before going ahead. Throughout this book, we’ll use jQuery Version 1.10.2 and jQuery UI Version 1.10.4 with the UI lightness theme. Note that the jQuery UI files also contain a copy of the jQuery source file.

Downloading the required files

To download jQuery, visit the download page on the jQuery website at http://jquery.com/download/.

[💡] Make sure that you download the correct version as per your requirements.
For jQuery UI, visit the download page at http://jqueryui.com/download/ to download the jQuery UI library.

Since we will cover all components throughout the book, download the full version of the library.

Using jQuery/jQuery UI libraries with a CDN
You might be aware that Content Delivery Networks (CDN) host many popular libraries. Since the browsers cache JavaScript files, your page will not have to load a JavaScript file again if it is referenced from a CDN and already cached in browser. You can link jQuery and jQuery UI among CDN’s many other libraries.

Make sure that you are connected to the Internet if you have referenced the libraries from the CDN in your pages.

Google, Microsoft, and some other companies provide CDN for jQuery, jQuery UI, and other libraries. Here are the links to pages for downloading these libraries:

- Google CDN can be downloaded from https://developers.google.com/speed/libraries/devguide#jquery.
- Microsoft CDN can be downloaded from http://www.asp.net/ajaxlibrary/cdn.ashx#Using_jQuery_from_the_CDN_21.
- CDNJS can be downloaded from https://cdnjs.com/. It is a helpful site where you can find many libraries and their different versions.

Setting up the folder structure for the JavaScript and CSS files
We will now set up the folder structure that we will use for all the chapters in this book. The steps are as follows:

1. In your document root, create a folder named MasteringjQueryUI. Then, create a folder for each chapter inside it.
2. For this chapter, create a new folder named Chapter1 inside MasteringjQueryUI and two more folders named js and css inside the Chapter1 folder.
Chapter 1

Now extract the jQuery UI files into a separate folder and go to its js folder. You will see three files: jQuery source file and full and minified versions of jQuery UI.

Copy the jQuery source file and any one version of jQuery UI source files and place them inside the js folder that you created inside Chapter1 of the MasteringjQueryUI folder.

Also, copy the ui-lightness folder from the css folder from the downloaded jQuery UI to the css folder of Chapter1.

Now we are ready to experiment with jQuery UI and create some informative and fun examples. Let's start our journey by creating the quiz application.

Creating the layout
In the newly created folder Chapter1, create a file named index.html and another .js file named quiz.js inside the js folder of Chapter1. The quiz.js file will contain all the code that we need to make the quiz functional.

Markup for the quiz page
Open the index.html file for editing using your favorite text editor, and write the following code in it:

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Designing a simple quiz application</title>
    <link rel="stylesheet" href="css/ui-lightness/jquery-ui-1.10.4.custom.min.css">
</head>
<body>
<div class="container">
    <span id="score"></span>
    <button id="reset" type="button">Reset</button>
    <div class="clear"></div>
    
    <hr/>
    
    <div id="leftCol">
        <ul id="source">
        </ul>
    </div>
    
    <div id="rightCol">
        <ul id="target">
        </ul>
    </div>
</div>
</body>
</html>
```
Designing a Simple Quiz Application

</div>

<div id="dialog-complete" title="Well Done!">
  <p><span class="ui-icon ui-icon-check"></span> Well done. You have completed the quiz successfully.</p>
</div>

<script src="js/jquery-1.10.2.js"></script>
<script src="js/jquery-ui-1.10.4.custom.min.js"></script>
<script src="js/quiz.js"></script>

</body>
</html>

Downloading the example code
You can download the example code files for all Packt books you have purchased from your account at http://www.packtpub.com. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

In the preceding code, inside the head section, we referenced the jQuery UI .css file. If you have placed the .css file elsewhere, make sure to correct the path accordingly. The path can either be an absolute path or a relative path.

Inside the body, we have a div element named container that wraps our entire markup. Inside container, we have created two span elements. The first span has the id value score and it will be used to show the score of the user. The second span has the id value reset and it will be used to reset the quiz to its initial state.

After this, we have to create two more div elements having the id value leftCol and rightCol, respectively. leftCol has an ul element with the id value source inside it. This ul element will contain the names of countries as list items. Similarly, rightCol has another ul element with the id value target inside it. It will have the names of capitals as list items.

After the container element, we have created yet another div element with the id value dialog-complete, which will be displayed once the user has completed the quiz successfully. Inside the dialog-complete element, we have placed a success message.

Finally, at the bottom of the page, reference the jQuery, jQuery UI, and quiz.js files.

Make sure that jQuery is included first before jQuery UI, and that any other JavaScript file or custom jQuery code is included or written.
Styling elements

We will also need some CSS styling to make the elements look good. In the head section of the index.html file, write the following code:

```html
<style type="text/css">
body{
  font-family:arial,verdana;
  font-size:12px;
  margin: 0px auto;
  width: 600px;
}

div.container{
  border: 1px solid #000;
  float:left;
  margin:0 auto;
  padding:10px;
  width: 100%;
}

#leftCol{
  float:left;
}

#rightCol{
  float:right;
}

ul{
  list-style:none;
  margin:0;
  padding:0;
  width:50%;
}

li{
  border:1px solid #000;
  font-weight:bold;
  margin:5px 0;
  padding:10px 0;
  text-align:center;
  width:175px;
}
```
In the preceding code, first we defined some basic styles for the body and the container elements. After that, the styles were defined for the ul element and its li items. These styles will display the list items in the form of individual boxes. CSS for the score and reset items follow next and finally some basic styling for the dialog elements.
Making the quiz functional

Our UI part is now complete and we can proceed to make the quiz functional. We will do this in a few steps. First of all, we will display the data on the page in two columns. Then, we will make the country names draggable. Finally, the list items with the capital names will be made droppable so that we can drop a country name inside a capital. We will also have to ensure that a droppable capital name accepts only the correct country name. Finally, the resetting logic will be built.

Displaying data on the page

Open the quiz.js file for editing and write the following code:

```javascript
$(document).ready(function()
{
    createQuizLayout();
});
```

On the document ready event we call a function named `createQuizLayout` which we need to define now.

```javascript
function createQuizLayout()
{
    //declare arrays of countries and their capitals.
    var countries = ["USA", "UK", "India", "Germany", "Turkey", "France", "Nepal", "Japan", "South Africa", "Maldives"];  
    var arrCountry = [];
    for(var i=0; i<countries.length; i++)
    {
        arrCountry.push('<li data-index="' + (i+1) + '">' + countries[i] + '</li>');</n    }  
     var arrCapital = [];
    for(var i=0; i<capitals.length; i++)
    {
        arrCapital.push('<li data-index="' + (i+1) + '">' + capitals[i] + '</li>');</n    }

    //shuffle the arrays
    arrCountry = shuffle(arrCountry);
```
Designing a Simple Quiz Application

```javascript
arrCapital = shuffle(arrCapital);
// once country and capital items are ready, we insert them into DOM
$('#source').html(arrCountry.join(''));
$('#target').html(arrCapital.join(''));
}
```

Here is what the preceding code does:

- We have defined two arrays named countries and capitals.
- The countries array contains names of 10 countries and the capitals array contains names of the capitals of the countries defined in the countries array. The names of capitals must be in the same order as their respective countries.
- Since we want to display the names of countries and capitals in a random order, we will create two arrays and fill them with list items and shuffle them.
- We started with country first. We declared an array named arrCountry. Then, we loop in the countries array and create a list item with the country name and push it into the arrCountry array.
- The same process is repeated for the capitals array.

An important point to note here is that we are giving a data attribute named index to each list item having a value from 1 to 10. Since we have both the countries and capital names in the same order, index will be used to match which country belongs to which capital.

After both arrays are populated, we will shuffle them so that the order of countries and capitals becomes random. For this, we will use a simple shuffle function from the website http://jsfromhell.com/array/shuffle. The shuffle function is defined as follows:

```javascript
function shuffle(o)
{
    for(var j, x, i = o.length; i; j = Math.floor(Math.random() * i), x = o[--i], o[i] = o[j], o[j] = x);
    return o;
}
```

After calling the shuffle function on both arrays arrCountry and arrCapital, the array elements are inserted in DOM after combining them into a single string using the JavaScript join function. The elements in the array arrCountry are inserted in ul with the id value source and those in the array arrCapital are inserted in ul with the id value target.
Open your browser and point it to the index.html file of the Chapter1 folder now. You will see a page similar to the one shown in the following screenshot:

![Image of a page with country and capital names]

If you reload the page, you will see that the order of countries and capitals changes each time. This is because shuffling creates a new order for items of both lists.

**Draggable country names**

To make the country names draggable, we will use the draggable component of jQuery UI. As the name suggests, the draggable component allows DOM components to be moved around using a mouse. To do this, go to the $(document).ready() section of our quiz.js file and call another function named initQuiz. The $(document).ready() callback function should look like this now:

```javascript
$(document).ready(function()
{
    createQuizLayout();
    initQuiz();
});
```
Now define the `initQuiz` function outside document ready handler as follows:

```javascript
function initQuiz()
{
    $('#source li').draggable(
    {
        revert : true,
        revertDuration: 200,
        cursor: "move"
    });
}
```

The preceding code calls the draggable method of the jQuery UI library. It is being called upon the li elements of the ul source, which means it will make all the list items draggable inside the source ul. Further, we are also giving the draggable method three options that we need for our application: revert, revertDuration, and cursor. Let's look at these in more detail:

- **revert**: This decides whether the element being dragged should revert to its original position or not. In our case, we will set it to `true`. We will drag a country name onto a capital name and revert it to its original position, that is, the country list. Another possible value for revert is `false`, which means it will stay at the place where it is when dragging stops. The values, valid and invalid, can also be provided (as strings) for the revert option. The value `valid` means the draggable object will revert only if the draggable object has been dropped on a droppable element. The value `invalid` means the draggable function will revert if the draggable object has not been dropped. Alternatively, a function can also be provided to revert. This is required in complex cases where we need to perform any manipulations. The return value for this function will decide if it will revert or not. If `true` is returned, the element will revert.

- **revertDuration**: This defines the duration for the revert option in milliseconds. The lower the value, the faster it will revert. This value is not considered if the revert option is set to `false`.

- **cursor**: This is the style of cursor while an element is being dragged.

Our draggable elements are ready now, and so it is time to make the capital names droppable and build the logic to match countries to their correct capitals.
Droppable capital names and scoring

In the previous section, we created an `initQuiz` function where we made our countries draggable. After the draggable code, write the following code to make the capitals droppable:

```javascript
var totalScore = 0;
$('#score').text(totalScore + ' points.');
$('#target li').droppable({
   accept: function(draggable)
   {
      if(parseInt(draggable.data('index'), 10) === parseInt($(this).data('index'), 10))
      {
         return true;
      }
      else
      {
         return false;
      }
   },
   drop: function( event, ui )
   {
      var that = $(this);
      that.addClass( "ui-state-highlight" ).html( 'Correct!' ).effect('bounce');
      that.droppable('disable');
      ui.draggable.addClass('correct ui-state-error');
      (ui.draggable).draggable('disable');
      totalScore++;
      $('#score').text(totalScore + ' points.');
      if($('li.correct').length == 10)
      {
         $( "#dialog-complete" ).dialog({
            resizable: false,
            modal: true,
         });
      }
   }
});
```
Designing a Simple Quiz Application

Now save the quiz.js file and refresh your browser. You will be able to drag the country names now. Drag a country name to its correct capital and you will see that the country will revert to its original position. The capital list item will show a bounce effect and its text will change to Correct!. Both the country and capital names will be disabled now. You will not be able to drag the country name as well. On the top left hand side, the page will show the score as **1 points**.

The screen will look like the following screenshot:
Try the drag and drop for all countries in the left-hand side list. When you have matched all countries correctly, you will see a dialog box and the page will look like the following screenshot:

So, a lot is happening in the preceding code. We will look at it step by step.

We defined a variable named `totalScore` and set it to 0. We also inserted the score inside the HTML element with the `id` value `score`. Each time the quiz starts, the score will be reset as well. After this, we call the `droppable` method of jQuery UI on the list items of `ul` with the `id` value `target` to make them ready to accept the draggable country elements.

We are using the `accept` option of the jQuery UI `draggable` method to check for the correct matches of country and capital, and we are using the `drop` event to change the UI and scoring.
Accepting a draggable element

The accept option of a droppable method defines which draggable element will be accepted by the droppable method when a draggable element is over it; either a jQuery selector or a function can be provided for this purpose. If a selector is given, only the draggable element matching that selector will be accepted by the droppable method. Since we want to match an individual country to its capital, it is better for us to use a function instead. The function will receive the current draggable element that is being dragged as a parameter. Inside the function, $(this) will refer to the current droppable element. The code is as follows:

```javascript
if(parseInt(draggable.data('index'), 10) == parseInt($(this).data('index'), 10))
{
    return true;
}
return false;
```

Since we have already defined data attributes for both countries and capitals, we can match those to check if the current draggable-droppable pair is a correct country-capital pair or not. If the indexes match, we return true; otherwise, we return false.

A return value true means the droppable method will accept the draggable element, and will allow the draggable element to be dropped in it.

The drop event

The drop event will receive a draggable element once it has been passed from the accept option. If the accept option returns false for any draggable element, then the drop event will not be called. In our case, this means we will only receive a country's draggable element and its corresponding capital's droppable element.

The callback function for the drop event receives two parameters: event and ui. Of these two, we are interested in the ui object. Among other values, it provides us with a reference to the draggable element that was dropped. To refer to the current droppable element where the draggable element is dropped, we have $(this) variable with us. The code is as follows:

```javascript
$( this ).addClass( "ui-state-highlight" ).html( 'Correct!' )
    .effect( 'bounce' );
$( this ).droppable('disable');
```

In the preceding code, we added the jQuery UI framework's CSS class ui-state-highlight to the current droppable element and then changed that list item's HTML content to Correct! and added the bounce effect to the droppable capital.
Since the droppable capital has been matched successfully with its country, we no longer need it as a droppable element. Hence, the preceding code uses the disable method of the droppable component to disable the droppable functionality.

The next two lines add CSS classes named correct and ui-state-error to the draggable method and then disable it. The code is as follows:

```javascript
ui.draggable.addClass('correct ui-state-error');
(ui.draggable).draggable('disable');
```

The correct class will be used to determine how many successful countries have been matched. The class ui-state-error is just for presentation purposes to make the successfully matched country name highlighted. Using the draggable disable method, we disable the specific draggable element as well, because it has been matched and we do not want it to be dragged again.

Since the drop event receives only the accepted draggable elements, we can safely increase the variable totalScore by 1 and insert the new value back to the DOM in the element score. This shows us the latest score each time a new match is made.

Finally, we count the number of list items in the countries’ column that have the CSS class named correct associated with them. Since we have 10 elements, if all the 10 list items have the CSS class correct attached to them, it means the quiz is complete. We then show a jQuery UI dialog component that we kept hidden in our HTML page initially.

### Resetting the quiz

If you were wondering why we created the functions createQuizLayout and initQuiz when we wrote the code without them, the answer is that we need to call them again. It is better not to repeat yourself. We can now reset the quiz without having to reload the page.

We have already created an element with id reset. Visit the $(document).ready() callback again and write the following code after those two function calls. The section will now look like this:

```javascript
$(document).ready(function()
{
  createQuizLayout();
  initQuiz();

  $('#reset').on('click', function()
  {
    $('#source li').draggable('destroy');
  });
}
```
Designing a Simple Quiz Application

```javascript
$('#target li').droppable('destroy');
createQuizLayout();
initQuiz();
});
});

We have an event handler registered at the click of the reset button. It is using the destroy method of jQuery UI on the draggable and droppable elements. The destroy method will remove the complete draggable and droppable functionality from respective elements. It will also remove any special CSS classes that jQuery UI might have applied earlier.

After bringing the page to its initial state, we call the createQuizLayout and initQuiz functions again, which will initialize our quiz once more.

Improving the quiz

This was a basic application to begin with. There can be many more enhancements to this quiz to make it more feature rich. You are encouraged to add more features to it and refactor the code as well.

Here are some ideas that you can start with:

- Remove successfully matched countries and capitals
- If you watch closely, we do not need the variable totalScore

![List items with the class correct are enough for calculating scores.]

- Allow negative scoring if the user drops the country in an incorrect capital

Summary

The draggable and droppable methods are important components of jQuery UI in order to make interactive applications. We explored a few options presented by these two components and created a simple quiz application in this process. We will see more options presented by these two components in the following chapters as well, where we will create a jigsaw puzzle game.
Where to buy this book

You can buy Mastering jQuery UI from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.