jQuery UI Form Widget Tutorial

In this tutorial, you'll start with a basic flight reservation form: it contains text fields, radio buttons, and the <button> element. By adding jQuery UI v
to the page and applying jQuery UI's form widgets to the page, we'll get a more v attractive, interactive, and functional form.

NOTE Tutorial files are located in your development directory.

1. **In your text editor, open the file *form.html* in the *chapter10* folder.**

   This file already contains a link to all of the jQuery and jQuery UI files and the $(document).ready() function (page 160). The HTML for the form on this page looks like this:

```
<form>
  <div>
    <label for="departure" class="label">Pick a departure date</label>
    <input type="text" id="departure" name="departure">
  </div>
  <div>
    <label for="airport" class="label">Find an airport</label>
    <input type="text" id="airport" name="airport">
  </div>
  <div>
    <label for="meal" class="label">Meal Options</label>
    <select name="meal" id="meal">
      <option>No meal</option>
      <option>Vegan</option>
      <option>Gluten Free</option>
```

```
      <option>Vegetarian</option>
      <option>Meat eater</option>
    </select>
  </div>
  <div id="bags">
    <p class="label">Number of bags to check</p>
    <input type="radio" id="none" name="bags" checked="checked">
    <label for="none">0</label>
    <input type="radio" id="one" name="bags">
    <label for="one">1</label>
    <input type="radio" id="two" name="bags">
    <label for="two">2</label>
  </div>
  <div id="seatTypes">
    <p class="label">Type of seat you'd like</p>
    <input type="checkbox" id="aisle" name="aisle">
    <label for="aisle">Aisle</label>
    <input type="checkbox" id="window" name="window">
    <label for="window">Window</label>
    <input type="checkbox" id="exit" name="exit">
    <label for="exit">Exit Row</label>
    <input type="checkbox" id="any" name="any">
    <label for="any">Any seat</label>
  </div>
  <div>
    <button id="next">Continue reservation process</button>
  </div>
</form>
```

The important form elements—the one's you'll apply jQuery UI to—are in bold. You'll start by adding a data picker to the first field. For the next step, note that the `<input>` element has the ID `departure`.

2. **Inside the `$(document).ready()` function, select the input element:**

```
$('#departure')
```

Now you need to apply the Datepicker widget.

3. **Type a period, followed by `datepicker();` so the code now looks like this:**

```
$('#departure').datepicker();
```

That's all that's needed to add a pop-up calendar date picker to the page. However, if you save the file and view it in a web browser now, you'll see that you can pick dates prior to today. Obviously, you can't exactly get on a flight that happened last week, so you want to make sure you can't make a reservation earlier than today's date.

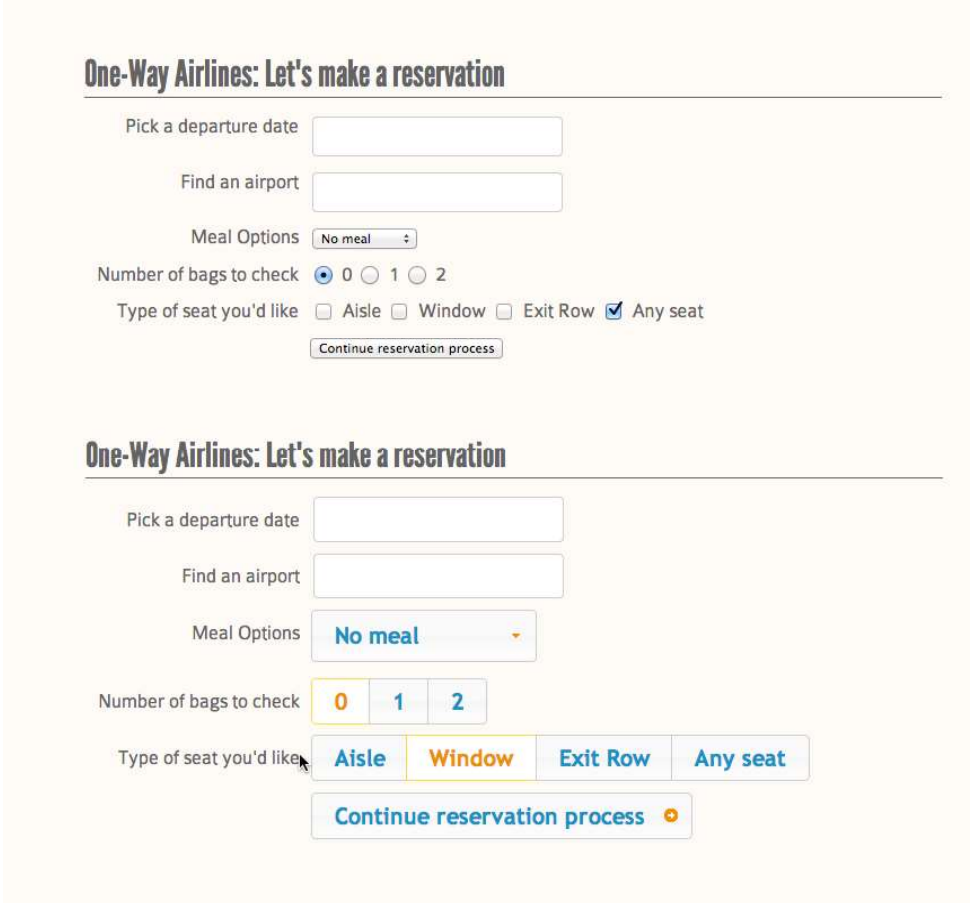

FIGURE 10-9
jQuery UI can make a drab form, match the look and feel of your entire application. It can transform form controls like select menus, radio buttons, and check boxes (which are normally styled by your browser) into attractive and attention-grabbing user interface elements.

4. **Click inside the `datepicker()` function and type a `{`. Press return twice and type `}` so that the code looks like this:**

```
$('#departure').datepicker({

});
```

The { } is an empty object literal, and you'll specify options for the widget by adding properties inside it. First, set the minimum date for making a reservation.

5. **On the blank line, inside the object literal type `minDate : 0`.**

The value for the `minDate` option indicates the number of days from today: 0 days from today *is* today; -7 days from today would be last week, and 7 days from today would be next week.

You can also specify the *maximum* date allowable by the date picker. In this case, the airline doesn't keep track of flights past 1 year from the current day, so you'll prevent someone from selecting dates later than that.

6. **Type a comma at the end of the last line you typed, hit Return, and type `maxDate : '+1y'` so the code looks like this:**

```
$('#departure').datepicker({
  minDate : 0,
  maxDate : '+1y'
});
```

Both `minDate` and `maxDate` let you use a number to indicate a specific number of days, but you can also use a string value that includes a number and a letter to indicate a length of time: `'+1y'` means 1 year in the future, `'-2w'` means 2 weeks in the past, and `'+1m +10d'` means 1 month and 10 days in the future.

7. **Save the page and preview it in a web browser. Click inside the "Pick a departure date" field.**

A pop-up calendar appears. Notice that you can't select a date earlier than today's date or more than one year from today's date (if you want to see for yourself, click the forward arrow to go forward 12 months).

Next, you'll use the Autocomplete widget to make selecting an airport easier. But first, take a look at the data source you're going to use. As mentioned on page 372, the `autocomplete()` function can accept either a JavaScript array of data, or data sent back from a server-side script running on a web server. To keep this example easier, you'll use a small amount of data stored in a separate JavaScript file.

8. **In a text editor, open the file *airports.js*.**

This is a simple JavaScript file containing an array assignment: it creates an array named `airports` filled with objects made up of two properties: `label` and `value`. The label will appear in the pop-up autocomplete menu, while the value will be written to the text field when the visitor selects an item from the list.

Obviously this file is not a complete list, just a small amount of data to try this widget out. To access the data in this file, you'll need to link it to the page.

9. **In your text editor, open the *form.html* file. Below the last `<script>` line—`<script src ="../_js/jquery-ui.min.js"></script>`—add one other script tag linking to the data file:**

```
<script src="airports.js"></script>
```

This line loads the external JavaScript file, and once it's loaded, the JavaScript will run. In this case, the file just creates an array full of data, which you can then use in the autocomplete widget.

10. **Add an empty line after the datepicker widget code you added earlier, and type:**

```
$('#airport').autocomplete({ source : 'airports.json'});
```

The text field for collecting an airport name has an ID of `airport`, so `$('#airport')` selects that field, and `.autocomplete()` applies the widget to that fields.

11. **Save the page and preview it in a web browser. Click inside the "Find an airport" field and type *Port*.**

    An autocomplete menu appears listing suggestions for airports that might be what you're looking for. Notice that the widget matches items that have "Port" *anywhere* in their label: "La Guardia Air*port*," for example.

12. **Click the "Portland International Airport, Portland, OR" item.**

    Notice that "PDX" appears in the text field. That's because you supplied an array of objects with labels and values. The label that appears in the drop-down menu was "Portland International Airport, Portland, OR", but the value added to the text field by the jQuery UI widget is "PDX," the airport code.

    Next, you'll change that drop-down menu into a thing of user interface beauty.

13. **Return to your text editor and the *form.html* file. Add the following code after the last line of code you added in step 10:**

```
$('#meal').selectmenu();
```

    You may have the urge to type more code, but that's all you need to do to transform the select menu. If you save and preview the page, however, you'll notice that the menu looks a bit weird. Actually, the first item on the menu isn't listed completely! Unless the labels in your select menu are very short, you'll need to set a width for this widget.

14. **Inside the parentheses for the `selectmenu()` function, type `{ width : 200 }` so the line of code looks like this:**

```
$('#meal').selectmenu( { width : 200 } );
```

    The `selectmenu` widget's `width` option (page 362) lets you set the width of the menu on the page. In this case, 200 means 200 pixels wide, though you can use other dimensions and types of measurement such as percentages or ems, as described on page 363. Save the page and check it out in a web browser. Who says programming's hard?

    Next, you'll convert the radio buttons and checkboxes into something that more closely resembles the rest of your form.

15. **In your text editor, add the following two lines after the code you added in the last step:**

```
$('#bags').buttonset();
$('#seatTypes').buttonset();
```

    The `buttonset()` function works for both radio buttons and checkboxes. The last step is to convert the `<button>` element into a jQuery UI button.

16. **Add one more line of code to your program:**

```
$('#next').button();
```

This converts the drab HTML button to something that looks a lot more like this form and matches the look of the radio buttons and checkboxes. You can also add jQuery UI icons to <button> elements, as described on page 367. Let's do that next.

17. **Inside the parentheses in the `button()` function add:**

```
{
  icons : {
    secondary : 'ui-icon-circle-arrow-e'
  }
}
```

The final JavaScript code for the page should look like this:

```
$(document).ready(function() {
  $('#departure').datepicker({
     minDate : 0,
     maxDate : '+1y'
  });
  $('#airport').autocomplete({ source : airports});
  $('#meal').selectmenu({width : 200});
  $('#bags').buttonset();
  $('#seatTypes').buttonset();
  $('#next').button({
      icons : {
          secondary : 'ui-icon-circle-arrow-e'
      }
  });
}); // end ready
```

In just a few lines of code, you've completely transformed the look and functionality of a web form.

18. **Save the page and preview it in a web browser.**

The page should look like the bottom image in Figure 10-9. If your page doesn't v look like this one, make sure the code you have matches the code in step 17. v You can also check the JavaScript console (page 18) for any errors; however, be v aware that jQuery often hides errors from the console, so it can make it harder to find errors in your jQuery code.

19. **Upload your completed form to your live server and test.**