LEARNING jQuery UI Library

#jquery-ui

Table of Contents

About

Chapter 1: Getting started with jQuery UI Library

Remarks

jQuery UI is a JavaScript UI library, built on top of jQuery, offering a set of user interface interactions, effects and widgets.

Versions

Version	Release Date
1.7.0	2009-03-06
1.7.1	2009-03-19
1.7.2	2009-06-12
1.7.4	2010-05-04
1.8.0	2010-03-23
1.8.1	2010-05-04
1.8.2	2010-06-07
1.8.4	2010-08-10
1.8.5	2010-09-17
1.8.6	2010-10-02
1.8.7	2010-12-10
1.8.8	2011-01-14
1.8.9	2011-01-21
1.8.10	2011-02-24
1.8.11	2011-03-18
1.8.12	2011-04-23
1.8.13	2011-05-17
1.8.14	2011-06-28

Version	Release Date
1.8.15	2011-08-08
1.8.16	2011-08-18
1.8.17	2012-01-10
1.8.18	2012-02-23
1.8.19	2012-04-17
1.8.20	2012-04-30
1.8.21	2012-06-05
1.8.22	2012-07-24
1.8.23	2012-08-15
1.8.24	2012-09-28
1.9.0	2012-10-08
1.9.1	2012-10-25
1.9.2	2012-11-23
1.10.0	2013-01-17
1.10.1	2013-02-15
1.10.2	2013-03-14
1.10.3	2013-05-03
1.10.4	2014-01-17
1.11.0	2014-06-26
1.11.1	2014-08-13
1.11.2	2014-10-16
1.11.3	2015-02-12
1.11.4	2015-03-11

Examples

Adding the jQuery UI script & basic usage

To get started with the jQuery UI library, you'll need to add the jQuery script, the jQuery UI script, and the jQuery UI stylesheet to your HTML.

First, download jQuery UI; choose the features you need on the download page. Unzip your download, and put `jquery-ui.css` and `jquery-ui.js` (and `jquery.js`) in a folder where you can use them from your HTML (e.g. with your other scripts and stylesheets.)

jQuery UI depends on jQuery, so remember to include `jquery.js` before `jquery-ui.js`.

```
<link rel="stylesheet" href="stylesheets/jquery-ui.css">
<script src="scripts/jquery.js"></script>
<script src="scripts/jquery-ui.js"></script>
```

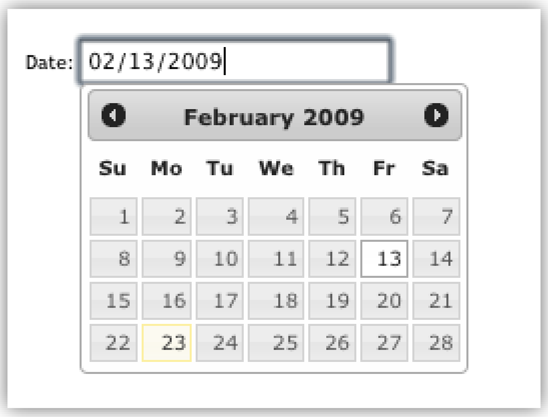
That's it! You can now use jQuery UI. For example, use the datepicker with the following HTML:

```
<input type="text" name="date" id="date">
```

Then use the following JavaScript:

```
$("#date").datepicker();
```

Which will get you a nice datepicker popup:



For more, see the **official "Getting started" gude**.

Setting up jQuery UI for the First Time Example

The jQuery UI framework helps to extend and increase the User Interface controls for jQuery JavaScript library.

When you wish to use jQuery UI, you will need to add these libraries to your HTML. A quick way to start is using the Content Delivery Network available code sources:

jQuery Libraries

```
https://code.jquery.com/jquery-3.1.0.js
https://code.jquery.com/ui/1.12.0/jquery-ui.js
```

You can choose many different themes for jQuery UI and even Roll your Own Theme. For this example, we will use 'Smoothness'. You add the theme via CSS.

jQuery UI CSS

```
https://code.jquery.com/ui/1.12.0/themes/smoothness/jquery-ui.css
```

Putting it all Together

When you have downloaded or selected your CDN, you will now want to add these libraries and style sheets to your HTML so that your web page can now make use of the jQuery and jQuery UI. The order in which you load the libraries is important. Call the jQuery library first, and then your jQuery UI library. Since jQuery UI extends jQuery, it must be called after. Your HTML may look something like the following.

```
<html>
<head>
  <title>My First UI</title>
  <link rel="stylesheet" href="https://code.jquery.com/ui/1.12.0/themes/smoothness/jquery-
ui.css">
  <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
  <script src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
  <script>
  $( function() {
    $( "#sortable" ).sortable();
    $( "#sortable" ).disableSelection();
  } );
  </script>
</head>
<body>

<ul id="sortable">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
  <li class="ui-state-default">Item 6</li>
  <li class="ui-state-default">Item 7</li>
</ul>

</body>
</html>
```

Read Getting started with jQuery UI Library online: https://riptutorial.com/jquery-ui/topic/513/getting-started-with-jquery-ui-library

Chapter 2: Accordion

Syntax

- $(function() { $("#selecter").accordion(); });
- $(function() { $("#selecter").accordion({ active: 2 }); });
- $(function() { $("#selecter").accordion({ animate: 200 }); });
- $(function() { $("#selecter").accordion({ collapsible: true }); });

Parameters

Parameter	Detail
active	Type Boolean or Integer, Boolean requires collapsible to be true
animate	Type Boolean, Number, String or Object
collapsible	Type Boolean
event	Type String
header	Type Selector element
heightStyle	Type String
icons	Type jQuery UI icon object

Remarks

More information can be found here: http://api.jqueryui.com/accordion/

Examples

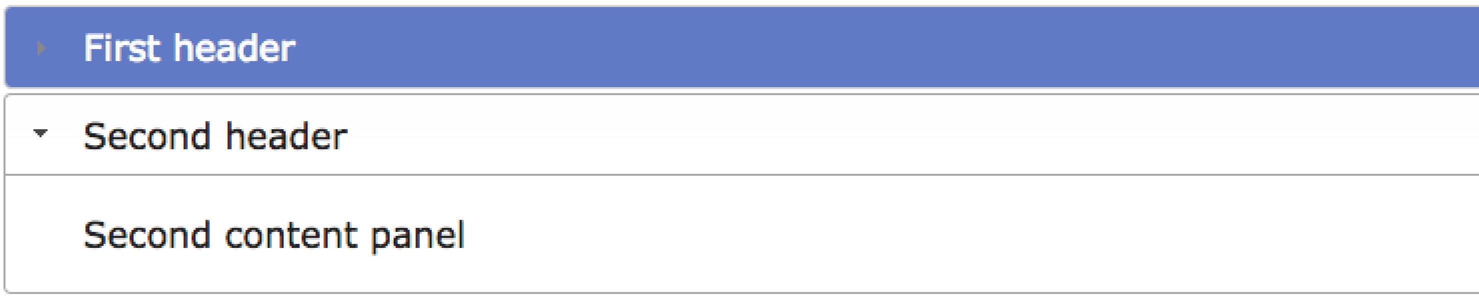
Accordion Basic Usage

To use an accordion, one must have headers and content inside the headers in their HTML. Then one must instantiate the `accordion()` method of jQuery UI.

```
<script>
$(function() {
    $( "#accordion" ).accordion();
});
</script>
```

In the HTML:

```
<div id="accordion">
    <h3>First header</h3>
        <div>First content panel</div>
    <h3>Second header</h3>
        <div>Second content panel</div>
</div>
```



Accordion destroy usage

```
$( "#accordion" ).accordion( "destroy" );
```

This will remove the accordion functionality completely and show default HTML removing all the jQuery-UI elements.

This method does not take any arguments.

Accordion disable Usage

```
$( "#accordion" ).accordion( "disable" );
```

This method will disable the accordion, i.e. the headers are not selectable making the content read only and static.

This method does not take any arguments.

Accordion enable Usage

```
$( "#accordion" ).accordion( "enable" );
```

This method will enable an accordion. This will enable a disabled accordion or simply do nothing on an already enabled accordion.

This method does not take any arguments.

Accordion option Usage

```
var options = $( "#accordion" ).accordion( "option" );
```

This will return a PlainObject giving all the options representing the selected accordion. This will contain all the values of the keys that are explained in the Parameters section.

This method takes parameters which are the basic optionNames explained in the parameter. One can set the options like this:

```
$( "#accordion" ).accordion( "option", "disabled", true );
```

Accordion refresh Usage

```
$( "#accordion" ).accordion( "refresh" );
```

This method recomputes the height of the accordion panels if headers or content was added or removed in the DOM.

Accordiong widget usage

```
var widget = $( "#accordion" ).accordion( "widget" );
```

This method returns a jQuery object containing the accordion.

Read Accordion online: https://riptutorial.com/jquery-ui/topic/630/accordion

Chapter 3: Autocomplete

Examples

Simple example

The Autocomplete widgets provides suggestions while you type into the field.

```
<script>
  $(document).ready(function() {
    var tags = ["ask","always", "all", "alright", "one", "foo", "blackberry",
"tweet","force9", "westerners", "sport"];
    $( "#tags" ).autocomplete({
      source: tags
    });
  });
</script>
<input type='text' title='Tags' id='tags' />
```

Read Autocomplete online: https://riptutorial.com/jquery-ui/topic/519/autocomplete

Chapter 4: Button

Syntax

- $(".selector").button();
- $(".selector").button({ disabled: true });
- $(".selector").button({ icons: { primary: "ui-icon-gear", secondary: "ui-icon-triangle-1-s" } });
- $(".selector").button({ label: "custom label" });
- $(".selector").button({ text: false });

Parameters

Parameter	Type - Details - Default
`disabled`	`Boolean` - Disables the button if set to true - `false`
`icons`	`Object` - Icons to display - { primary: null, secondary: `null` }
`label`	`String` - Text to show in the button - `null`
`text`	`Boolean` - Whether to show the label - `true`

Examples

Basic usage

Create an input (or button, or anchor) html element and call `button()` method of jQuery UI.

```
<script>
$(function() {
    $( "#myButton" ).button();
});
</script>
```

HTML

```
<input type="button" value="A button" id="myButton">
```

Read Button online: https://riptutorial.com/jquery-ui/topic/4600/button

Chapter 5: Datepicker

Examples

Initialization

The **datepicker** is used to show an interactive date selector which is tied to a standard form input field. It makes selecting of date for input tasks very easy and also it is also highly configurable.

Any input field can be bound with jquery-ui datepicker by the input field's selector (id,class etc.) using **datepicker()** method like this -

```
<input type="text" id="datepicker">
<script>
    $("#datepicker").datepicker();
</script>
```

Live demo is here.

Setting Minimum and Maximum dates for a datepicker

```
<script>
$( ".inclas").datepicker({
  minDate: new Date(2007, 1 - 1, 1)
  maxDate: new Date(2008, 1 - 1, 1)
});
</script>

<input type ="text" id="datepick" class="inclas">
```

Show week of the year

The following code will show week of the year number on the left side of the datepicker. By default the week start on Monday, but it can be customized using `firstDay` option. The first week of the year contains the first Thursday of the year, following the ISO 8601 definition.

```
<input type="text" id="datepicker">
<script>
    $("#datepicker").datepicker({
        showWeek: true
    });
</script>
```

Set a custom date format

Default date format: "mm/dd/yy"

The following example shows how you can set the date format in initialization with the dateFormat option.

```
<input type="text" id="datepicker">
<script>
    $("#datepicker").datepicker({
        dateFormat: "yy-mm-dd"
    });
</script>
```

The following example shows how you can set the date format after initialization with the dateFormat option.

```
<input type="text" id="datepicker">
<script>
    $("#datepicker").datepicker( "option", "dateFormat", "yy-mm-dd" );
</script>
```

You can use combinations of the following:

```
d - day of month (no leading zero)
dd - day of month (two digit)
o - day of the year (no leading zeros)
oo - day of the year (three digit)
D - day name short
DD - day name long
m - month of year (no leading zero)
mm - month of year (two digit)
M - month name short
MM - month name long
y - year (two digit)
yy - year (four digit)
@ - Unix timestamp (ms since 01/01/1970)
! - Windows ticks (100ns since 01/01/0001)
'...' - literal text
'' - single quote
anything else - literal text
```

Or predefined standard:

```
ATOM - 'yy-mm-dd' (Same as RFC 3339/ISO 8601)
COOKIE - 'D, dd M yy'
ISO_8601 - 'yy-mm-dd'
RFC_822 - 'D, d M y' (See RFC 822)
RFC_850 - 'DD, dd-M-y' (See RFC 850)
RFC_1036 - 'D, d M y' (See RFC 1036)
RFC_1123 - 'D, d M yy' (See RFC 1123)
RFC_2822 - 'D, d M yy' (See RFC 2822)
RSS - 'D, d M y' (Same as RFC 822)
TICKS - '!'
TIMESTAMP - '@'
W3C - 'yy-mm-dd' (Same as ISO 8601)
```

A default date format can be applied to all datepickers using the following example:

```
<script>
```

```
    $.datepicker.setDefaults({
        dateFormat: "yy-mm-dd"
    });
</script>
```

Show month and year dropdown

jQuery datepicker has two options to allow displaying dropdowns for month and year selection. These options make navigation through large timeframes easier.

```
<input type="text" id="datepicker">
<script>
    $("#datepicker").datepicker({
        changeMonth: true, // shows months dropdown
        changeYear: true   // shows years dropdown
    });
</script>
```

Read Datepicker online: https://riptutorial.com/jquery-ui/topic/520/datepicker

Chapter 6: Dialog

Syntax

- $(".selector").dialog("option", "disabled"); // Option Getter, specific
- $(".selector").dialog("option"); // Option Getter all
- $(".selector").dialog("option", "disabled", true); // Option Setter, specific
- $(".selector").dialog("option", { disabled: true }); // Option Setter, multiple
- $(".selector").dialog("close"); // Triggers close
- $(".selector").dialog({ close: function() {} }); // Close overloading
- $(".selector").on("dialogclose", function(event, ui) {}); // Close overloading

Parameters

Parameter	Description
Options	
appendTo	(Selector) [Default: `"body"`] Which element the dialog (and overlay, if modal) should be appended to.
autoOpen	(Boolean) [Default: `true`] If set to true, the dialog will automatically open upon initialization. If false, the dialog will stay hidden until the open() method is called.
buttons	(Object/Array) Specifies which buttons should be displayed on the dialog. The context of the callback is the dialog element; if you need access to the button, it is available as the target of the event object.
closeOnEscape	(Boolean) [Default: `true`] Specifies whether the dialog should close when it has focus and the user presses the escape (ESC) key.
closeText	(String) [Default: `"close"`] Specifies the text for the close button. Note that the close text is visibly hidden when using a standard theme.
dialogClass	(String) The specified class name(s) will be added to the dialog, for additional theming.
draggable	(Boolean) [Default: `true`] If set to `true`, the dialog will be draggable by the title bar. Requires the jQuery UI Draggable widget to be included.
height	(Number/String) [Default: `"auto"`] The height of the dialog.
hide	(Bool/Num/Str/Obj) If and how to animate the hiding of the dialog.
maxHeight	(Number) [Default: `false`] The maximum height to which the dialog can be

Parameter	Description
	resized, in pixels.
maxWidth	(Number) [Default: `false`] The maximum width to which the dialog can be resized, in pixels.
minHeight	(Number) [Default: `150`] The minimum height to which the dialog can be resized, in pixels.
minWidth	(Number) [Default: `150`] The minimum width to which the dialog can be resized, in pixels.
modal	(Boolean) [Default: `false`] If set to true, the dialog will have modal behavior; other items on the page will be disabled, i.e., cannot be interacted with. Modal dialogs create an overlay below the dialog but above other page elements.
position	(Object) [Default: `{ my: "center", at: "center", of: window }`] Specifies where the dialog should be displayed when opened. The dialog will handle collisions such that as much of the dialog is visible as possible.
resizable	(Boolean) [Default: `true`] If set to true, the dialog will be resizable. Requires the jQuery UI Resizable widget to be included.
show	(Bool/Num/Str/Obj) If and how to animate the showing of the dialog.
title	(String) Specifies the title of the dialog. If the value is null, the title attribute on the dialog source element will be used.
width	(Number) [Default: `300`] The width of the dialog, in pixels.
Methods	
close	Closes the dialog.
destroy	Removes the dialog functionality completely. This will return the element back to its pre-init state.
instance	Retrieves the dialog's instance object. If the element does not have an associated instance, undefined is returned.
isOpen	Whether the dialog is currently open.
moveToTop	Moves the dialog to the top of the dialog stack.
open	Opens the dialog.
option	Gets the value currently associated with the specified `optionName`.
option	Gets an object containing key/value pairs representing the current dialog

Parameter	Description
	options hash.
option	Sets one or more options for the dialog.
widget	Returns a jQuery object containing the generated wrapper.
Extension Points	
_allowInteraction	(event) Modal dialogs do not allow users to interact with elements behind the dialog. This can be problematic for elements that are not children of the dialog, but are absolutely positioned to appear as though they are. The `_allowInteraction()` method determines whether the user should be allowed to interact with a given target element; therefore, it can be used to whitelist elements that are not children of the dialog but you want users to be able to use.
Events	
beforeClose	(event, ui) Triggered when a dialog is about to close. If canceled, the dialog will not close.
close	(event, ui) Triggered when the dialog is closed.
create	(event, ui) Triggered when the dialog is created.
drag	(event, ui { position, offset }) Triggered while the dialog is being dragged.
dragStart	(event, ui { position, offset }) Triggered when the user starts dragging the dialog.
dragStop	(event, ui { position, offset }) Triggered after the dialog has been dragged.
focus	(event, ui) Triggered when the dialog gains focus.
open	(event, ui) Triggered when the dialog is opened.
resize	(event, ui { originalPosition, position, originalSize, size }) Triggered while the dialog is being resized.
resizeStart	(event, ui { originalPosition, position, originalSize, size }) Triggered while the dialog is being resized.
resizeStop	(event, ui { originalPosition, position, originalSize, size }) Triggered while the dialog is being resized.

Remarks

Parameter Source: http://api.jqueryui.com/dialog/

Examples

Simple Example

Dialog is a window which is overlay positioned within the viewport.

```
<script>
  $(function() {
    $( "#dialog" ).dialog();
  });
</script>
<div id="dialog" title="Basic dialog">
  <p>This is the default dialog which is useful for displaying information. The dialog window
can be moved, resized and closed with the 'x' icon.</p>
</div>
```

Open dialog when event occurs

Usually we want to separate the creation of the dialog from its appearance. Then three steps are needed.

1. Create base element

```
<div id="dialog" title="Basic dialog">
  <p>This is the default dialog which is useful for displaying information. The dialog window
can be moved, resized and closed with the 'x' icon.</p>
</div>
```

2. Make it a dialog, note the `autoOpen: false` option that ensures that it will be closed at first

```
$( "#dialog" ).dialog({
  autoOpen: false
});
```

3. Open it when needed, like on a button click

```
$( "#dialog" ).dialog( "open" );
```

Complex Example - jQuery UI Dynamicly Create Dialog

Generally, dialog relies on a `div` within the HTML. Sometimes you may want to create a dialog from scratch, programmatically. Here is an example of a complex modal dialog created dynamically with interactive functions.

HTML

```
<div id="users-contain" class="ui-widget">
```

```
  <h1>Existing Users:</h1>
  <table id="users" class="ui-widget ui-widget-content">
    <thead>
      <tr class="ui-widget-header ">
        <th>Name</th>
        <th>Email</th>
        <th>Password</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>John Doe</td>
        <td>john.doe@example.com</td>
        <td>johndoe1</td>
      </tr>
    </tbody>
  </table>
</div>
<button id="create-user">Create new user</button>
```

CSS

```
label,
input {
  display: block;
}

input.text {
  margin-bottom: 12px;
  width: 95%;
  padding: .4em;
}

fieldset {
  padding: 0;
  border: 0;
  margin-top: 25px;
}

h1 {
  font-size: 1.2em;
  margin: .6em 0;
}

div#users-contain {
  width: 350px;
  margin: 20px 0;
}

div#users-contain table {
  margin: 1em 0;
  border-collapse: collapse;
  width: 100%;
}

div#users-contain table td,
div#users-contain table th {
  border: 1px solid #eee;
  padding: .6em 10px;
  text-align: left;
}

.ui-dialog .ui-state-error {
  padding: .3em;
}

.validateTips {
  border: 1px solid transparent;
  padding: 0.3em;
}
```

jQuery

```
$(function() {
  // Define variables for the dialog, form and a regular expression used to verify email
addresses in the form
  var dialog, form,
    emailRegex = /^[a-zA-Z0-9.!#$%&'*+\/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-
9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$/;

  // Function to update tips when an issue in the form is detected
  // t = text to enter as the tip
  function updateTips(t) {
    tips
      .text(t)
      .addClass("ui-state-highlight");
    setTimeout(function() {
      tips.removeClass("ui-state-highlight", 1500);
    }, 500);
  }

  // Function to check the length of text entered into a field
  // o = object reference (object), n = name of field (string), min = minimum number of
characters (int), max = maximum number of characters (int)
  function checkLength(o, n, min, max) {
    if (o.val().length > max || o.val().length < min) {
      o.addClass("ui-state-error");
      updateTips("Length of " + n + " must be between " +
        min + " and " + max + ".");
      return false;
    } else {
      return true;
    }
  }

  // Function to perform regular expression check of text entered in field
  // o = object reference (object), regexp = regular expression reference (RegExp Object), n =
name of field
  function checkRegexp(o, regexp, n) {
    if (!(regexp.test(o.val()))) {
      o.addClass("ui-state-error");
      updateTips(n);
      return false;
    } else {
      return true;
    }
  }

  //Function called when form is submitted that will check all the form fields. If all fields
have text and all the text meets the requirements, the data is collected and added back to the
```

```
table.
  function addUser() {
    var valid = true;
    allFields.removeClass("ui-state-error");

    valid = valid && checkLength(name, "username", 3, 16);
    valid = valid && checkLength(email, "email", 6, 80);
    valid = valid && checkLength(password, "password", 5, 16);

    valid = valid && checkRegexp(name, /^[a-z]([0-9a-z_\s])+$/i, "Username may consist of a-z,
0-9, underscores, spaces and must begin with a letter.");
    valid = valid && checkRegexp(email, emailRegex, "eg. ui@jquery.com");
    valid = valid && checkRegexp(password, /^([0-9a-zA-Z])+$/, "Password field only allow : a-
z 0-9");

    if (valid) {
      $("#users tbody").append("<tr>" +
        "<td>" + name.val() + "</td>" +
        "<td>" + email.val() + "</td>" +
        "<td>" + password.val() + "</td>" +
        "</tr>");
      dialog.dialog("close");
    }
    return valid;
  }

  // Creation of the dialog object
  dialog = $("<div>", {
    id: "dialog-form",
    title: "Create New User"
  }).dialog({
    autoOpen: false,
    height: 400,
    width: 350,
    modal: true,
    buttons: {
      "Create an account": addUser,
      Cancel: function() {
        dialog.dialog("close");
      }
    },
    close: function() {
      form[0].reset();
      allFields.removeClass("ui-state-error");
    }
  });

  // Adding elements to the dialog to be shown
  dialog.html("<p class='validateTips'>All form fields are required.</p>")

  // Creation of the form object to be shown inside the dialog
  form = $("<form>").submit(function(e) {
    e.preventDefault();
    addUser();
  }).appendTo(dialog);

  // Adding elements to the form, fieldset and fields
  form.append($("<fieldset>"));
  var markup = "";
  markup += "<label for='name'>Name</label>\r\n";
  markup += "<input type='text' name='name' id='name' value='Jane Smith' class='text ui-
```

```
widget-content ui-corner-all'>";
  markup += "<label for='email'>Email</label><input type='text' name='email' id='email'
value='jane@smith.com' class='text ui-widget-content ui-corner-all'>\r\n";
  markup += "<label for='password'>Password</label><input type='password' name='password'
id='password' value='xxxxxxx' class='text ui-widget-content ui-corner-all'>\r\n";
  markup += "<input type='submit' tabindex='-1' style='position:absolute; top:-1000px'>\r\n";

  // Assigning our fields HTML markup to the fieldset
  form.find("fieldset").html(markup);

  // Assigning variables to be used for easy reference, post creation and amendment of dynamic
objects
  var name = $("#name"),
    email = $("#email"),
    password = $("#password"),
    allFields = $([]).add(name).add(email).add(password),
    tips = $(".validateTips");

  // Override the click event of the button to launch our dynamic dialog
  $("#create-user").button().on("click", function() {
    dialog.dialog("open");
  });
});
```

Working example for reference: https://jsfiddle.net/Twisty/LqjuxLu1/

Creating a Dialog with Tabbed Titlebar

Occasionally, we may want to display dialogs with more than one pane of content. jQuery UI offers tabs that can be used in tandem with a dialog to make this possible. While it may be more common to have tabs within a dialog's content container, this example will demonstrate how to make a list of tabs the titlebar of the dialog.

HTML

```
<button id="openButton">
  Open Dialog
</button>
<div id="dialog" style="display:none">
  <div class="ui-tabs">
    <ul>
      <li><a href="#tab_1">Tab 1</a></li>
      <li><a href="#tab_2">Tab 2</a></li>
    </ul>
    <div id="tab_1">
      <p>Tab 1 content...</p>
    </div>
    <div id="tab_2">
      <p>Tab 2 content...</p>
    </div>
  </div>
</div>
```

jQuery

```
$(document).ready(function() {
  // Options to pass to the jQuery UI Dialog
  var options = {
    position: {
      my: "left top",
      at: "left top",
      of: window
    },
    autoOpen: false
  };

  /* Initialization */
  // Initialize the dialog
  var dialog = $("#dialog").dialog(options);

  // Initialize the tabs
  var tabs = $(".ui-tabs").tabs();

  /* Gather Elements Before Rearrangement */
  var closeButton = dialog.siblings(".ui-dialog-titlebar").find(".ui-dialog-titlebar-close");
  var initialTitlebar = dialog.siblings(".ui-dialog-titlebar");

  // Find the list of tabs to make the titlebar, add the ui-dialog-titlebar class, and append
the close button
  var tabbedTitlebar = dialog.find(".ui-tabs ul:first").addClass("ui-dialog-
titlebar").append(closeButton);

  /* Arranging */
  // Remove the initialTitlebar
  $(initialTitlebar).remove();

  // Create a new .ui-tabs container for the tabbedTitlebar
  var tabbedTitlebarContainer = $("<div>", {
    class: "ui-tabs"
  }).append(tabbedTitlebar);

  // Prepend the tabbedTitlebarContainer to the dialog container
  dialog.parents(".ui-dialog").prepend(tabbedTitlebarContainer);

  /* Show the Dialog */
  dialog.dialog("open");

  var openButton = $("#openButton").button().click(function() {
    dialog.dialog("open");
  });
});
```

Working example for reference: https://jsfiddle.net/5x4zz681/

Dialog with no close button

If you like to show the dialog without the close button (i.e. the x button in the upper-right corner of the dialog), perhaps because you want to force the user to select one of options or buttons in the dialog itself:

1- Give your dialog a CSS class:

```
$("#selector").dialog({
    closeOnEscape: false,
    dialogClass: "dialog-no-close",
});
```

2- Hide the close button using this CSS:

```
.dialog-no-close .ui-dialog-titlebar-close {display: none; }
```

Note: If you want to hide the entire title bar, use this CSS instead:

```
.dialog-no-close .ui-dialog-titlebar {display: none; }
```

Alternatively, you can hide the close button in the dialog's initialization code:

```
$("#selector").dialog({
    closeOnEscape: false,
    open: function(event, ui) {
        $(".ui-dialog-titlebar-close", $(this).parent()).hide();
    }
});
```

Read Dialog online: https://riptutorial.com/jquery-ui/topic/521/dialog

Chapter 7: Draggable

Examples

Simple Example

Enable draggable functionality on any DOM element.

```
<script>
  $(function() {
    $( "#draggable" ).draggable();
  });
</script>
<div id="draggable" class="ui-widget-content">
  <p>Drag me around</p>
</div>
```

Draggable with handle

You can use any element as an handle to drag another element around:

```
<script>
  $(function() {
      $( "#draggable" ).draggable({
         handle: ".handle"
      });
  });
</script>
<div id="draggable">
    <span class="handle">Handle</span>
    <div>Content</div>
</div>
```

Fiddle

Read Draggable online: https://riptutorial.com/jquery-ui/topic/522/draggable

Chapter 8: Icons

Syntax

- .ui-icon-{icon type}-{icon sub description}-{direction}

Remarks

The icons are also integrated into a number of jQuery UI's widgets, such as accordion, button, menu.

Examples

Basic usage

For a thick arrow pointing north in a span, add classes `ui-icon` and `ui-icon-arrowthick-1-n`:

```
<span class="ui-icon ui-icon-arrowthick-1-n"></span>
```

For a triangle pointing south in a span, add classes `ui-icon` and `ui-icon-triangle-1-s`:

```
<span class="ui-icon ui-icon-triangle-1-s"></span>
```

Full list of availliable items here https://api.jqueryui.com/theming/icons/

Read Icons online: https://riptutorial.com/jquery-ui/topic/4633/icons

Chapter 9: jQuery UI Rotatable Plug-in

Parameters

Parameter	Details
handle	url to a custom image for the handle
angle	the starting rotation for the element.
rotationCenterX	position about which the element will be rotated
rotationCenterY	position about which the element will be rotated
step	an angle in degrees that the rotation will snap to if the shift key is held.
snap	snaps to step in degrees.
start	triggered when rotation starts
stop	triggered when rotation stops
rotate	triggered when object is being rotated
wheelRotate	enable/disable mouse wheel to rotate element.

Examples

Initial Usage Example

jquery-ui-rotatable is a plugin for jQuery UI that works in a similar way to Draggable and Resizable, without being as full-featured. By default, it puts a small rotation icon in the bottom left of whatever element you want to make rotatable.

```
<html>
  <head>
    <title>My Rotatable</title>
    <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-
ui.css">
    <link rel="stylesheet"
href="//cdn.jsdelivr.net/jquery.ui.rotatable/1.0.1/jquery.ui.rotatable.css">
    <script src="http://code.jquery.com/jquery-1.11.3.js"></script>
    <script src="http://code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
    <script
src="//cdn.jsdelivr.net/jquery.ui.rotatable/1.0.1/jquery.ui.rotatable.min.js"></script>
    <script>
    $(function(){
      $('#target').rotatable();
    });
    </script>
  </head>
  <body>
  <div id="target">Rotate me!</div>
  </body>
</html>
```

Read jQuery UI Rotatable Plug-in online: https://riptutorial.com/jquery-ui/topic/1806/jquery-ui-rotatable-plug-in

Chapter 10: jquery ui sortable

Examples

jQuery UI Sortable - Drop Placeholder

This example of the Sortable using a Placeholder is common usage. Sortable is applied to a group of DOM elements, allowing the user to move items around in the list via Drag'n Drop style actions.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>jQuery UI Sortable - Drop Placeholder</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.12.0/themes/base/jquery-ui.css">
  <style>
  #sortable {
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 60%;
  }
  #sortable li {
    margin: 0 5px 5px 5px;
    padding: 5px;
    font-size: 1.2em;
    height: 1.5em;
  }
  html>body #sortable li {
    height: 1.5em; line-height: 1.2em;
  }
  .ui-state-highlight {
    height: 1.5em;
    line-height: 1.2em;
  }
  </style>
  <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
  <script src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
  <script>
  $( function() {
    $( "#sortable" ).sortable({
      placeholder: "ui-state-highlight"
    }).disableSelection();
  });
  </script>
</head>
<body>

<ul id="sortable">
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
  <li class="ui-state-default">Item 6</li>
  <li class="ui-state-default">Item 7</li>
</ul>

</body>
</html>
```

Read jquery ui sortable online: https://riptutorial.com/jquery-ui/topic/1226/jquery-ui-sortable

Chapter 11: Slider

Examples

Simple Example

A slider control uses draggable handles to select numeric values. Below is an example of a basic slider initialization:

```
<script>
  $(function() {
    $( "#slider" ).slider();
  });
</script>
<div id="slider"></div>
```

Range Slider

Range sliders provide 2 draggable handles to select numeric values. The slider's initialization must provide a `range` option set to `true` to create a range slider:

```
<script>
  $(function() {
    $( "#range-slider" ).slider({
        range: true
    });
  });
</script>
<div id="range-slider"></div>
```

Initializing Values and Value Limits

A slider element can have its value set on initialization by providing a `value` option. This option is a number:

```
$( "#slider" ).slider({
    value: 5
});
```

A range slider can also have its values set in this way by providing a `values` option. This option is an array of numbers:

```
$( "#range-slider" ).slider({
    range: true,
    values: [5, 25]
});
```

In addition to providing initial values, the minimum value, maximum value, and handle interval can be defined with the `min`, `max`, and `step` options, respectively:

```
$( "#range-slider" ).slider({
    range: true,
    min: 0,    // The lowest possible value will be 0
    max: 100,  // The highest possible value will be 100
    step: 5,   // The slider handles will lock in at intervals of 5
    values: [0, 100]
});
```

Using the Slide Event

The slider provides an event called `slide` that will trigger **whenever the mouse moves during a slider handle drag**. This function holds a reference to the slide `event` and a reference to the slider `ui` object. The `ui` object holds a jQuery object for the handle being moved and the value(s) of the slider.

Single-Handle Slider:

```
var value;

$( "#slider" ).slider({
    slide: function(event, ui) {
      value = ui.value;
    }
});
```

Range Slider:

```
var lowValue;
var highValue;

$( "#range-slider" ).slider({
    range: true,
    slide: function(event, ui) {
      lowValue = ui.values[0];
      highValue = ui.values[1];
    }
});
```

Note: The `slide` event is intended to respond to active mouse motion and will not trigger if the slider values are changed programmatically. To react to these events, use the `change` event.

Setting Values and the Change Event

The slider provides an event called `change` that will trigger **after the mouse completes a slider handle drag or if the value(s) have been changed programmatically**. This function holds a reference to the slide `event` and a reference to the slider `ui` object. The `ui` object holds a jQuery object for the handle being moved and the value(s) of the slider.

One example could be having to display new information after a slider's values have been updated by another element's event. Let's use a `select` element for demonstration where the value of the slider is programmatically set in when the value of the `select` changes:

HTML

```
<select id="setting">
  <option value="1">Low</option>
  <option value="2">Medium</option>
  <option value="3">High</option>
</select>

<div id="slider"></div>

<div id="display-value"></div>
```

JavaScript

```
$(function() {
  $( "#slider" ).slider({
    min: 0,
    max: 11,
    // This will trigger when the value is programmatically changed
    change: function(event, ui) {
        $( "#display-value" ).text(ui.value);
    }
  });

  $( "#setting" ).change(function () {
    switch ($(this).val()) {
      case "1":
        $( "#slider" ).slider( "value", 3 );  // Sets the value of a slider programmatically
        break;
      case "2":
        $( "#slider" ).slider( "value", 7 );  // Sets the value of a slider programmatically
        break;
      case "3":
        $( "#slider" ).slider( "value", 11 ); // Sets the value of a slider programmatically
        break;
    }
  });
});
```

Note: It's in these circumstances that the `slide` event would not trigger and the `change` event is needed. However, if elements need to react to the slider values changing as the handle is being dragged, the `slide` event will be necessary.

Read Slider online: https://riptutorial.com/jquery-ui/topic/3206/slider

Chapter 12: Sortable

Syntax

- $("#sortable").sortable({ /*Options Here*/ }); //Initialise Sortable
- $("#sortable").sortable("option", "option_name", option_value); //Set option outside initialiser
- var value = $("#sortable").sortable("option", "option_name"); //Gets the value of an option

Parameters

Parameter	Description
Options	
appendTo	(jQuery, Element, Selector, String) [Default: "parent"] The element that the helper is added to
axis	(String) [Default: false] The directions that the item can be dragged (x or y)
cancel	(Selector) [Default: "input,textarea,button,select,option"] Doesnt start sorting if you start on an element matching the selector
classes	(Object) [Default: {}] Specify additional classes to add to the sortables elements when the structural classes are added. ({ui-sortable-helper: custom_class})
connectWith	(Selector) [Default: false] Allows to items from one sortable to be dragged to another
containment	(Element, Selector, String) [Default: false] The element that items are constrained to
cursor	(String) [Default: "auto"] Defines the type of cursor to be shown when sorting
cursorAt	(Object) [Default: false] Defines the position that the helper looks like its being moved from
disabled	(Boolean) [Default: false] Disables the sorting if true
dropOnEmpty	(Boolean) [Default: true] If false items from this sortable can not be placed in empty sortables
forceHelperSize	(Boolean) [Default: false] Forces the helper to have a size

Parameter	Description
forcePlaceholderSize	(Boolean) [Default: false] Forces the placeholder to have a size
grid	(Array) [Default: false] Defines a grid to snap the helper to ([x,y])
handle	(Selector, Element) [Default: false] Defines elements that sorting can start on. Opposite to cancel
helper	(String, Function) [Default: "original"] String "original" or "clone", or function that returns the element to be used as the helper.
items	(Selector) [Default: "> *"] Defines the items that should be sortable
opacity	(Number 0.01 to 1) [Default: false] Defines the opacity for the helper
placeholder	(String) [Default: false] Defines a class or classes to be applied to the placeholder
revert	(Boolean, Number) [Default: false] The time that it takes the helper to slide into its new position
scroll	(Boolean) [Default: true] Whether to scroll when at edges of the page
scrollSensitivity	(Number) [Default: 20] Defines how close to the edge of the page the cursor needs to be to start scrolling
scrollSpeed	(Number) [Default: 20] The speed at which to scroll
tolerance	(String) [Default: "intersect"] Defines which mode to use when calculating when one item is over another ("intersect" or "pointer")
zIndex	(Integer) [Default: 1000] Defines the z-index of the helper when sorting
Methods	
cancel()	Cancels the current sort and returns the elements back to their position before the sort started
destroy()	Removes the sortable functionality and returns the element to its state pre initialisation
disable()	Disables the sortable
enable()	Enables the sortable
instance()	Returns the sortables instance object
option()	Gets key value pairs of all the options for the sortable

Parameter	Description
option(String)	Gets the value of an option
option(String, Any)	Sets the value of the option specified by the String
option(Object)	Sets one or more options with the object being key value pairs of options
refresh()	Refreshes the sortable options reloading all sortable items. This causes new items to be recognised
refreshPositions()	Rrefreses the cached positions of the sortable items
serialize(Object)	Serializes the items ids (by default) into a string that can be submitted or appended to a url Object options: {key: sets the key in the serialized string, attribute:[Default "id"] sets the attribute to look at, expression:[Default: "/(.+)-=_/"] regex to split the attribute in to key value pairs}
toArray(Object)	Serializes the sortable items id into an array. The object can contain a parameter attribute which has the attribute to put into the array default is id
widget()	Returns a jQuery object of the sortable element
Events	
activate(event, ui)	Triggered when connected list, every connected list on drag start receives it
beforeStop(event, ui)	Triggers before sorting stops when the helper has being shiftered to the same position as the placeholder
change(event, ui)	Triggered when elements change position ie. when the placeholder moves
create(event, ui)	Triggered when sortable is created
deactivate(event, ui)	Triggered when sorting stops. This goes to all connected lists as well
out(event, ui)	Triggered when the item is moved out of the sortable list
over(event, ui)	Triggered when the item is moved into a sortable list
receive(event, ui)	Triggered when an item from a connected list has being dropped into another one. Target is the receiving list
remove(event, ui)	Triggered when an item from a connected list has being dropped into another one. Target is the giving list

Parameter	Description
sort(event, ui)	Triggered during sorting
start(event, ui)	Triggered when sorting starts
stop(event, ui)	Triggered when sorting stops
update(event, ui)	Triggered when sorting stops and the DOM position has being updated

Remarks

Official Documentation here

Examples

Simple Example

Take any list and add an identifier to the outer wrapper (`ul`, `div`)

```
<ul id="sortable">
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
</ul>
```

In your jquery:

```
$(function(){
    $('#sortable').sortable({
        //pass all options in here
    });
});
```

This will allow all the `li` in the `#sortable` wrapper to be dragged and dropped in the list

Sortable Grid with flex layout

This used the flex layout with the sortable to create a grid of responsive boxes that can be moved around by dragging and dropping.

HTML

```
<div id="sortable">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
</div>
```

JS

```
$(function(){
    $('#sortable').sortable({
        //pass all options in here
    });
});
```

CSS

```
#sortable{
    width: 500px;
    display: flex;
    flex-wrap: wrap;
}
#sortable div {
    margin: 10px;
    background-color: #f00;
    flex-basis: 100px;
    height: 100px;
}
```

Stationary Items when dragging

This example uses a class on the placeholder to turn it into a line and make it take up no room.

HTML

```
<div id="sortable">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
</div>
```

JS

```
$("#sortable").sortable({
    placeholder: 'placeholder',
    helper: 'clone',
    start: function(event, ui){
        ui.item.show();
    }
});
```

CSS

```
#sortable div{
  background-color: #f00;
  width: 50px;
  height: 50px;
  margin: 10px;
  padding: 0px;
}
#sortable div.placeholder{
  height: 4px;
  margin: -7px 10px;
}
```

Sortable - Animate revert of unaccepted item

Working Example: https://jsfiddle.net/Twisty/4f5yh3pa/7/

Cancelling and Reverting a sortable is not strongly documented. The helps show how moving an item from one list to another connected list can be conditionally cancelled. by default, this is not animated by sortable, this example includes an animation.

Result: List #2 only accepts items that have a class of `acceptable`. Both lists can be sorted naturally otherwise.

HTML

```
<div class="ui-widget">
  <ul id="sortable1" class="connectedSortable">
    <li class="ui-state-default acceptable">Item 1</li>
    <li class="ui-state-default">Item 2</li>
    <li class="ui-state-default">Item 3</li>
    <li class="ui-state-default">Item 4</li>
    <li class="ui-state-default">Item 5</li>
  </ul>
  <ul id="sortable2" class="connectedSortable">
    <li class="ui-state-default">Item 6</li>
    <li class="ui-state-default acceptable">Item 7</li>
  </ul>
</div>
```

CSS

```
.ui-widget {
  position: relative;
}

.connectedSortable {
  border: 1px solid #eee;
  width: 142px;
  min-height: 20px;
  list-style-type: none;
  margin: 0;
  padding: 5px 0 0 0;
  float: left;
  margin-right: 10px;
}

#sortable1 {
  background: #fff;
}
```

```
#sortable2 {
  background: #999;
}

.connectedSortable li {
  margin: 0 5px 5px 5px;
  padding: 5px;
  font-size: 1.2em;
  width: 120px;
}
```

JavaScript

```
$(function() {
  $(".connectedSortable").sortable({
    connectWith: ".connectedSortable",
    receive: function(e, ui) {
      var $self = $(this);
      var $item = ui.item;
      var $sender = ui.sender;
      // Restrict condition to only one specific list if desired
      if ($(e.target).attr("id") == "sortable2") {
        if ($item.hasClass("acceptable")) {
          // Item Accepted
          console.log($self.attr("id") + " accepted item from: #" + $sender.attr("id") + " > "
+ $item.text());
        } else {
          // Item Rejected
          console.log($self.attr("id") + " rejected item from: #" + $sender.attr("id") + " > "
+ $item.text());
          // Animate the return of the items position
          $item.css("position", "absolute").animate(ui.originalPosition, "slow", function() {
            // Return the items position control to it's parent
            $item.css("position", "inherit");
            // Cancel the sortable action to return it to it's origin
            $sender.sortable("cancel");
          });
        }
      }
    }
  }).disableSelection();
});
```

Read Sortable online: https://riptutorial.com/jquery-ui/topic/1415/sortable

Chapter 13: Spinner

Syntax

- $("#id").spinner();
- $("#id").spinner({min:0,max:100,step:5,spin:function(event, ui) {}});

Parameters

Parameters	Detail
min	Minimum value
max	Maximum value
step	How much the value increases by on spinner click, can be decimal
spin	Can be used to check the spinner value, `ui.value` and do something

Remarks

Official Example

Official Documentation

Examples

Basic Example

Makes entering numbers a bit handier by showing a set of arrows on the right side of the `input`.

HTML

```
<link rel="stylesheet" href="//code.jquery.com/ui/1.12.0/themes/base/jquery-ui.css">
<script src="https://code.jquery.com/jquery-1.12.4.js"></script>
<script src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
<script src="/resources/demos/external/jquery-mousewheel/jquery.mousewheel.js"></script>
<script>
  $( function() {
    var spinner = $( "#spinner" ).spinner();
  } );
</script>

<input id="spinner" name="value">
```

Read Spinner online: https://riptutorial.com/jquery-ui/topic/6637/spinner

Credits

S. No	Chapters	Contributors
1	Getting started with jQuery UI Library	Community, J F, jkdev, JonasCz, Twisty
2	Accordion	Dipen Shah, Madalina Taina
3	Autocomplete	J F
4	Button	Theodore K.
5	Datepicker	cteski, J F, ni8mr, Nishant123, Peter Tirrell, Pradeep, smdsgn, VincenzoC
6	Dialog	J F, Jonathan Michalik, Racil Hilan, Theodore K., Twisty
7	Draggable	empiric, J F
8	Icons	Theodore K.
9	jQuery UI Rotatable Plug-in	Twisty
10	jquery ui sortable	Andrew Mcghie, Twisty
11	Slider	Jonathan Michalik
12	Sortable	Alon Eitan, Andrew Mcghie, M B, Twisty
13	Spinner	Alon Eitan, Andrew Mcghie, depperm