



## Lab 10: jQuery

### Learning Objectives

After completion of this lab, **you should be able** to

- Work collaboratively as a **pair programming** team
- Use **jQuery** files from the **CDN**
- Use **jQuery Validation Plugin** to simplify validation
- Add **custom error messages** with JavaScript
- Apply **Styling** with jQuery Themes
- Use the **jQuery UI** plugin
- **To Receive Credit**

---

### Work collaboratively as a pair programming team

CS 250 in-class labs will be done using **pair programming**. Your partner for today's lab is listed

in the table below:

**Hebeler 204**  
Grader: John Wright II

<b>Team 1</b> Abundiz, Sergio Chandler, Alan	<b>Team 2</b> Bajwa, Deepinder Dickerson, Andrew	<b>Team 3</b> Burley, Jonathan Juarez, Adrian	<b>Team 4</b> Byars, Frank Plitkins, Kristofer	<b>Team 5</b> Carpenter, Daniel Prescott, Brandon
<b>Team 6</b> Crockett, Jordan Canada, Justin	<b>Team 7</b> Hansen, Christopher Ahmady, Temourshah	<b>Team 8</b> Kinkade, Kyle Belfiglio, Alexander	<b>Team 9</b> Porter Jr, Anthony Burton, Henry	<b>Team 10</b> Rozelle, William Strom, Brandt
<b>Fill in:</b> Taing, Pokuy				

You may wish to review basic [pair programming](#) guidelines before you begin.

- One team member (the **driver**) has control of the keyboard/mouse and actively implements the program
- The other team member (the **navigator**) continuously observes the work of the driver to identify tactical defects (such as syntactic and spelling errors, etc.) and also thinks strategically about the direction of the work

You should **change roles** about every ten minutes during lab.

# Use jQuery files from the CDN

Today's lab will use the [jQuery](#) Core JavaScript Library and two of its plug-ins: [Validation](#) and [User Interface](#) (**UI**).

Rather than downloading the JavaScript files from the website, you should **hotlink** to the jQuery files from the CDN (Common Delivery Network) inside the `<head>` section of your HTML file.

```
<script type="text/javascript"

src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>
<script type="text/javascript"

src="http://ajax.microsoft.com/ajax/jquery.validate/1.7/jquery.validate.min.js">
</script>
<script type="text/javascript"
    src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.4/jquery-
ui.min.js">
</script>
```

You will only need to  
create **two files** for this

## Lab 10: jQuery

lab, stored in folder  
`U:\labs\lab10\`

- `lab10.html` file to contain a XHTML form
- `lab10.js` which will contain custom JavaScript code to work with the jQuery library files

To save time (and promote consistency), the `lab10.html` **XHTML code** is provided for you to **copy and paste** into your `lab10.html` file

1. Add the **hotlinks** to the jQuery files in the `<head>` section
2. Add the `script` element in the `<head>` section that links to your `lab10.js` JavaScript file
3. Add a [file header comment](#) to your

### Contact Details

Name

Company

Email

Telephone

Website

Submit Contact Details

### Project Details

Type of Project

Select Project Type ▾

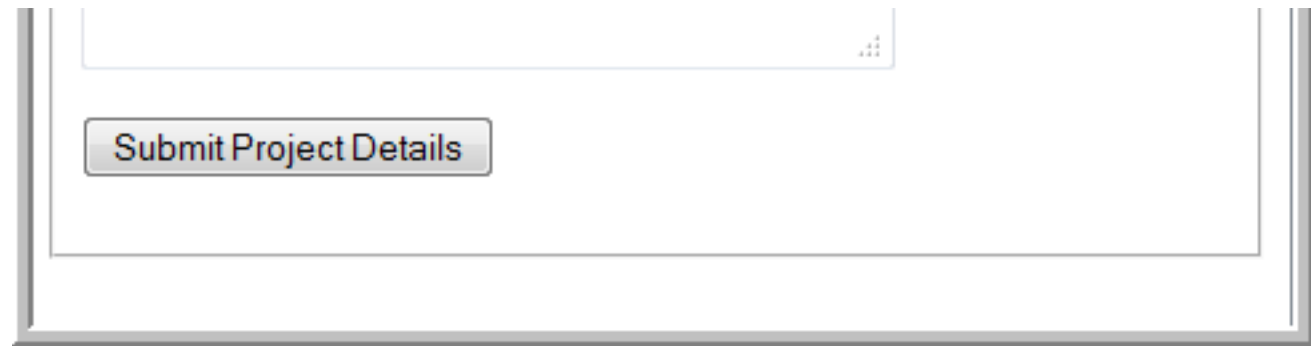
Projected Completion Date

Estimated Budget

### Additional Details

JavaScript file

Before going further,  
make sure your  
`lab10.html` page  
**validates** to XHTML 1.0  
Strict and contains **no** FAE  
**accessibility** issues.



---

## Use jQuery Validation Plugin to simplify validation

The dominant pattern used with jQuery is to **wait until** the document's DOM is loaded into the browser before selecting any DOM element. Otherwise, you may end up missing key DOM elements when selecting with jQuery selectors.

```
// Called when the document's DOM is ready
$(document).ready(function()
{
    // JavaScript code for anonymous function goes here
}
); // end document.ready event handler
```

To **add validation rules** to the Contact Details form

1. Use jQuery (\$) to get the **form** element with the unique **id**, **contactDetailsForm**
2. Add the **validate** event handler method with two **rules**
  - Apply the **required** rule (true) to the input element whose **name** attribute value is set to **"name"**
  - Apply the **required** rule (true) to the input element whose **name** attribute value is set to **"email"**

```
// Called when the document's DOM is ready
$(document).ready(function()
{
    // Contact Details Form validation
    $('#contactDetailsForm').validate(
    {
        rules : {
            name : {
                required : true,
            },
            email : {
                required : true,
            },
        }, // end rules
    }
```

```
}); // end validate #contactDetailsForm  
}); // end document.ready event handler
```

3. **Confirm** in the browser that your required field validation code is working for the name and email fields
  - Experiment with the web page in the browser to determine the behavior of the jQuery validation methods
4. **Add** the JavaScript code to make the **website** field **required** as well
5. Leave the company and telephone fields **optional**

The screenshot shows a web form titled 'Contact Details'. It contains five input fields: 'Name', 'Company', 'Email', 'Telephone', and 'Website'. The 'Name', 'Email', and 'Website' fields have the text 'This field is required.' displayed to their right. The 'Company' and 'Telephone' fields are empty. At the bottom of the form is a button labeled 'Submit Contact Details'.

In addition to the build in `required()` method, jQuery Validation plugin provides the **standard** [list of built-in validation methods](#)

```
email : {  
    required : true,  
    email : true,  
},
```

6. Add jQuery **email()** and **url()** validation methods for email and website fields.

Contact Details

Name  This field is required.

Company

Email aa  Please enter a valid email address.

Telephone aa

Website aa  Please enter a valid URL.

Furthermore, we can add **our own validation methods** that work cleanly with jQuery's Validation plugin.

For example, the following code adds the validation method **allLetters** which requires that the string parameter contains one or more letters (a-zA-Z) or the blank character (Note the use of a regular expression).

```
$.validator.addMethod('allLetters', function(value) {  
    return value.match(/^ [a-zA-Z ]+$/);  
});
```



Since this is the equivalent of declaring a JavaScript method (without invoking it), it can go anywhere inside `lab10.js` file, **outside** the scope of the event handler for **ready**.

We can use the custom validation method to **validate** that customer names consist of all letters by adding it to the rules for the name field.

```
$.validator.addMethod('allLetters', function(value) {  
    return value.match(/^([a-zA-Z ]+)$/);  
});  
  
$(document).ready(function()  
{  
    // Contact Details Form validation  
    $('#contactDetailsForm').validate(  
    {  
        rules : {  
            name : {  
                required : true,  
                allLetters : true,  
            },  
        },  
    });  
});
```

## 7. Add the **allLetters**

method and the validation **rule** for the customer name field.

8. Write a

Name 11 Warning: No message defined for name

Company CWU

Email aa@bbb.ccc

Telephone aa Warning: No message defined for telephone

Website http://bbb.aaa.com

Submit Contact Details

**phoneNumber** method that uses a regular expression to validate the customer's telephone field

- Use method **allLetters** as the model for **phoneNumber**
- Require phone numbers to be in the form ddd-ddd-dddd or ddd-dddd

9. Add the validation **rule** to the **telephone** field

Note: Since the telephone number is optional, the validation rule should accept an empty string (^\$) **or** the valid number

```
value.match(/(^$)|(you figure out this)/);
```

---

## Add custom error messages with JavaScript

To **customize the error messages** produced by the jQuery Validation plugin, we can specify the string to use as the custom error messages

**Note:** you can also override the default error message should you desire.

```
$('#contactDetailsForm').validate(  
  {  
    rules : {  
      name : {  
        required : true,  
      },  
      // ...  
    }, // end rules  
  
    // Custom error messages  
    messages : {  
      name : {  
        required : 'Your full name is required',  
        allLetters : 'Only letters or space allowed in name  
field',  
      },  
    }, // end messages  
  }); // end validate #contactDetailsForm
```

1. Add custom error

Contact Details

Name

11

Only letters or space allowed in name field

Company

CWU

Email

aa@bbb.ccc

Telephone

aa

Telephone must be in form ddd-ddd-dddd or ddd-dddd

Website

http://bbb.aaa.com

Submit Contact Details

messages for **allLetters** and **phoneNumber**.

2. Override any default error messages you would like to re-word

Should one desire, it is possible to add "valid" **confirmation** labels to fields that validate correctly

```
}, // end messages
```

```
// generate valid confirmation messages
```

```
success: function(label) {
```

```
        label.html("valid").addClass('valid');  
    },  
}); // end validate #contactDetailsForm
```

Contact Details

Name

Your name is required

Company

Email

Email must be in form aa@bbb.ccc

Telephone

valid

Website

valid

Submit Contact Details

---

## Apply Styling with jQuery Themes

Examine the **generated** HTML code (use Firefox View Source Chart) and determine how the jQuery Validation plugin inserts the error messages into the form elements. Note that it also inserts **class** attribute values into the code elements as well.

We can take advantage of this by applying our own style rules

```
<style type="text/css">
label.error {background:url("error.jpg") no-repeat;
             margin-left:5px; padding-left:25px;
             color:#CC0000; font-weight:bold;
}
input.error {background-color:#ffff99;
}
label.valid {background:url("valid.jpg") no-repeat;
             color : #FFFFFF
}
</style>
```

1. The rules use two images. Download and save the images in your lab10 folder
  - [error.jpg](#)
  - [valid.jpg](#)
2. Add the style rules to `lab10.html` **head** section

Contact Details

Name  X Your name is required


Company

Email  apple X Email must be in form aa@bbb.ccc

Telephone  509-963-1435 ✓

- For this lab, use document-level CSS rules, applied to the page



Website  

- Document-level style rules are often called embedded CSS, versus external *linked* CSS or inline CSS

There are a number of open-source [jQuery Themes](#) available for working with jQuery plugins. In particular, Microsoft makes available over a dozen themes from their CDN

We will use the **Redmond** jQuery UI Theme CSS file hosted at the Microsoft CDN

```
<link type="text/css" rel="Stylesheet"
      href="http://ajax.microsoft.com/ajax/jquery.ui/1.8.5/themes/redmond/jquery-
      ui.css" />
```

1. Add the hotlink to the Redmond CSS style sheet to the <head> section of lab10.html
  - Do **not** download the CSS file, link to it at the Microsoft site

Inside the document.ready method, add the JavaScript code to **select** the form element and its child elements and add **classes** to the forms' elements

```
$(document).ready(function()
```

```
{  
    // add styling classes to the forms' elements  
    $("form").addClass("ui-widget");  
    $("form fieldset").addClass("ui-widget-content ui-corner-  
all");  
    $("form legend").addClass("ui-widget-header ui-corner-all");  
    $("form input").addClass("ui-widget-content");  
    $("form textarea").addClass("ui-widget-content");  
  
    // Identify the two buttons and let jQuery add classes, etc.  
    $("#submitContactDetails").button();  
    $("#submitProjectDetails").button();  
}
```

Examine the generated code to determine what is happening



## Contact Details

Name  ✖ Only letters or space allowed in name field

Company

Email  ✖ Email must be in form aa@bbb.ccc

Telephone  ✔

Website  ✖ This field is required.

Submit Contact Details

## Use the jQuery UI plugin

**Note:** the following is **not** included on the [lab's scoring rubric](#).

In the interest of time, we will only use one UI widget from the jQuery UI plugin: the **datepicker**

```
$(document).ready(function()
```

```
{  
    $('#completionDate').datepicker();  
}
```

1. Add the JavaScript code above that selects the element with the unique **id** of

Project Details

Type of Project 

Select Project Type ▾

Projected Completion Date

Estimated Budget

Additional Details

Submit Project Details

February 2011

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28					

**completionDate** and makes it a jQuery **datepicker**

## 2. Test your Completion Date field

Add the JavaScript code to **validate** the **#projectDetailsForm** using the jQuery Validate plugin

**Tip:** Add a new validate method after the validate #contactDetailsForm method ends

```
}); // end validate #contactDetailsForm

// validate project details form
$('#projectDetailsForm').validate(
{
    rules : {
```

## Project Details

Type of Project  ✖ Please select a project type

Projected Completion Date  ✖ This field is required.

Estimated Budget  ✖ Please enter a valid number.

Additional Details

Submit Project Details

1. Make Project Type required  
**Hint:** use the **min()** validation method with a custom error message
2. Make Project Completion Date required
3. Let Estimated Budget be optional, but validate for a valid number if entered  
**Hint:** use the **number()** validation method

## To Receive Credit

Labs are graded by the student teaching assistant using the [lab's scoring rubric](#) [PDF].

Pair programming teams will receive the scoring rubric sheet at the start of lab. Write both names on the sheet to turn in the sheet in when you finish. Name 1 should be the student who saved the pair work in their CS 250 account.

- If you **finish during lab**, turn in to your grader the **Lab's scoring rubric sheet**.
  - If you have **not** yet satisfied all criteria to the level of 4, you need to continue working on the lab outside of class time
  - Your saved solution should be stored in the CS 250 account listed under Name 1
  - Your solution will be checked by the grader using the scoring rubric and both students will receive the same score for the lab assignment
- If you are **unable** to complete the solution during the lab period
  - Leave lab with both students having a copy of the partially completed solution
  - Agree to finish the lab together (establish a time) or independently
  - Turn in the [lab's scoring rubric](#) [PDF] at the **start of next Lab**
    - Use one rubric for teams finishing together or two rubrics for students finishing independently
    - Keep track of and include the **total completion time** (rounded to closest half-hour) it took to complete the lab assignment and include the time on the rubric
    - Write a score **0 . . 4** in the rubric's **self assessment column** representing your completion status
  - Make sure your work is stored in your CS 250 account in folder  
`U:\htdocs\labs\lab10\`

Lab 10 is due at the start of lab of the next Lab. **No** late lab assignments will be accepted without prior approval. Your lowest lab score will be dropped.