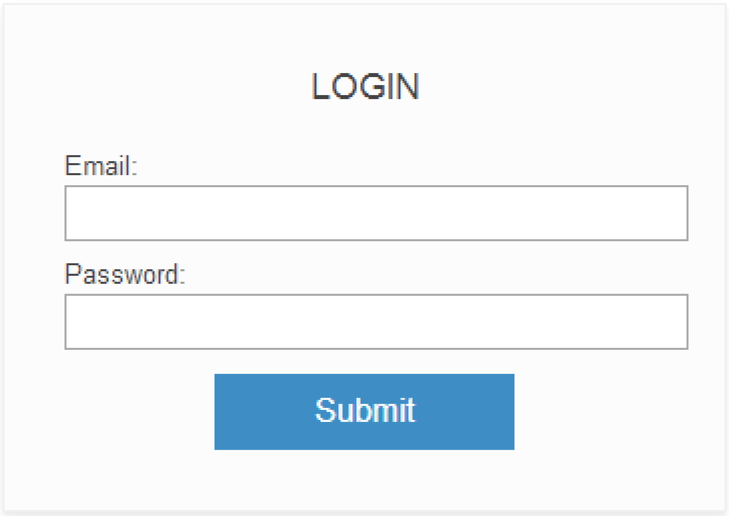
"The Web Design Workshop" Decal – Spring 2014
Programming Handout 9

Goals:

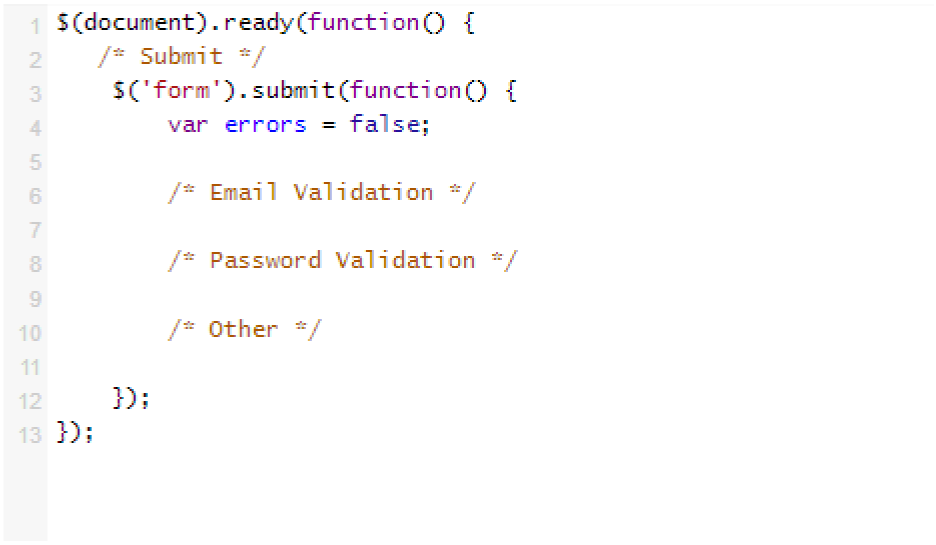
1. Form Validation

1 – Form Validation

1. Go to http://jsfiddle.net/5RP63/3/ . You will see this in the bottom right:



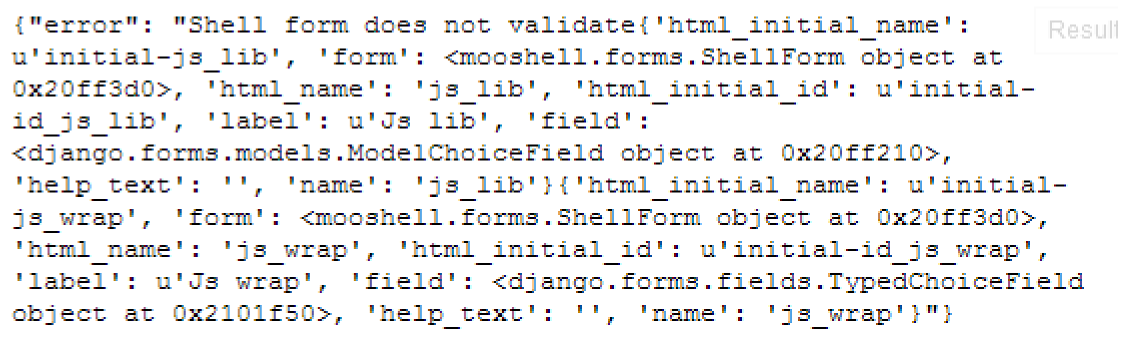
And this is the template jQuery on the bottom left:



```
$(document).ready(function() {
   /* Submit */
    $('form').submit(function() {
        var errors = false;

        /* Email Validation */

        /* Password Validation */

        /* Other */

    });
});
```

2. Notice we have a new **event listener** called **.submit()**, attached to our form element. This only triggers when we click on the *<input type="submit" value="Submit">* element.

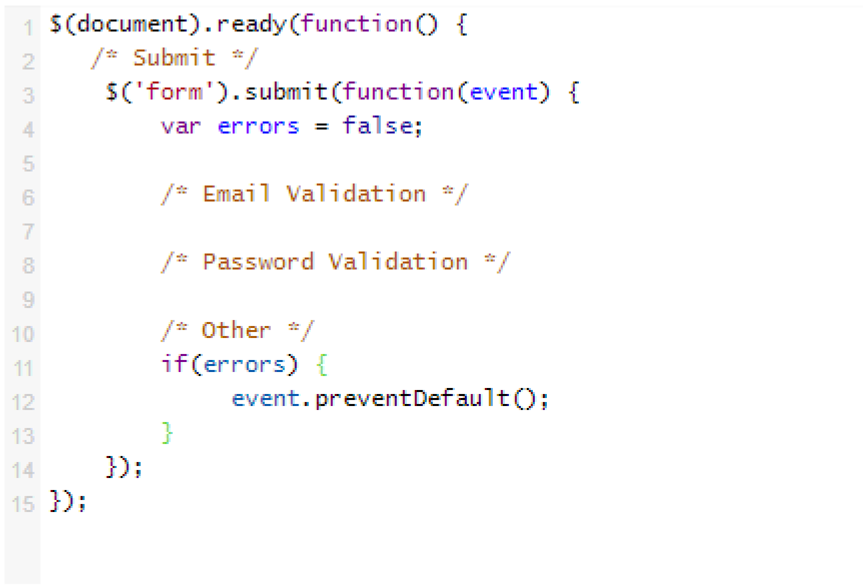
But, if you just click on Submit button on the bottom right, you will see this:



```
{"error": "Shell form does not validate{'html_initial_name':
u'initial-js_lib', 'form': <mooshell.forms.ShellForm object at
0x20ff3d0>, 'html_name': 'js_lib', 'html_initial_id': u'initial-
id_js_lib', 'label': u'Js lib', 'field':
<django.forms.models.ModelChoiceField object at 0x20ff210>,
'help_text': '', 'name': 'js_lib'}{'html_initial_name': u'initial-
js_wrap', 'form': <mooshell.forms.ShellForm object at 0x20ff3d0>,
'html_name': 'js_wrap', 'html_initial_id': u'initial-id_js_wrap',
'label': u'Js wrap', 'field': <django.forms.fields.TypedChoiceField
object at 0x2101f50>, 'help_text': '', 'name': 'js_wrap'}"}
```

This is because if you submit a form, it will do *something* on the backend (which we will not go over). Before we even submit a form though, we want to make sure that there are no errors on the form (on the front end). If there is, we want to **prevent the default action** of the submit event (which is to send the form to the backend). Thus, by preventing the form from submitting the above image won't happen. And, we learned how to prevent default in the last lecture.

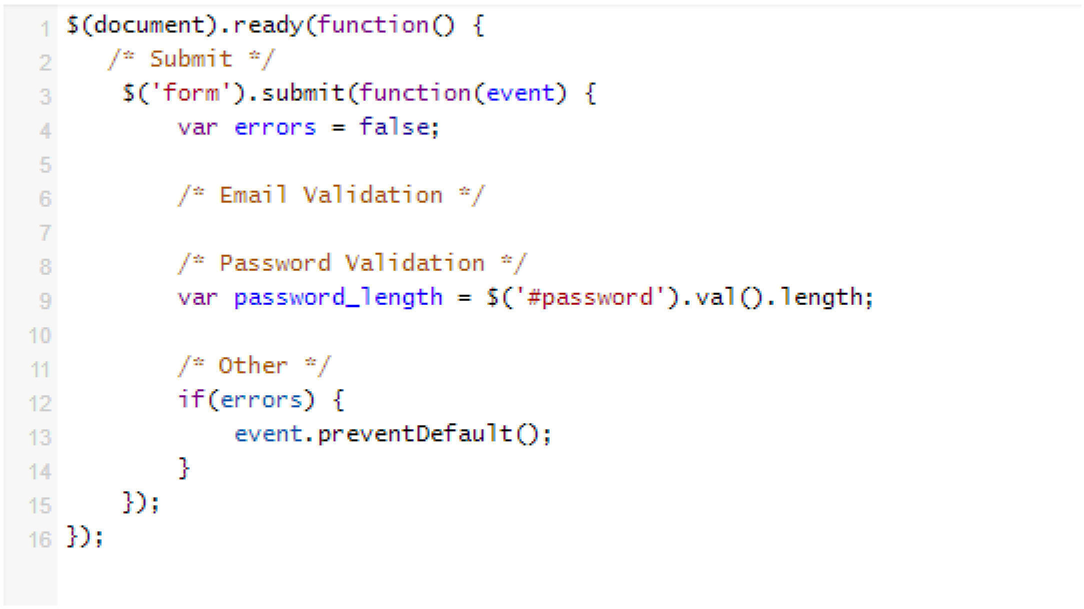
Pass in an **event object** to the submit event listener, and then inside the event, if there are **errors**, prevent the default action of the form (sending the form) like so:



```
$(document).ready(function() {
   /* Submit */
    $('form').submit(function(event) {
        var errors = false;

        /* Email Validation */

        /* Password Validation */

        /* Other */
        if(errors) {
             event.preventDefault();
        }
    });
});
```

3. Notice that our variable **errors** won't be true just yet because we haven't set it to true.

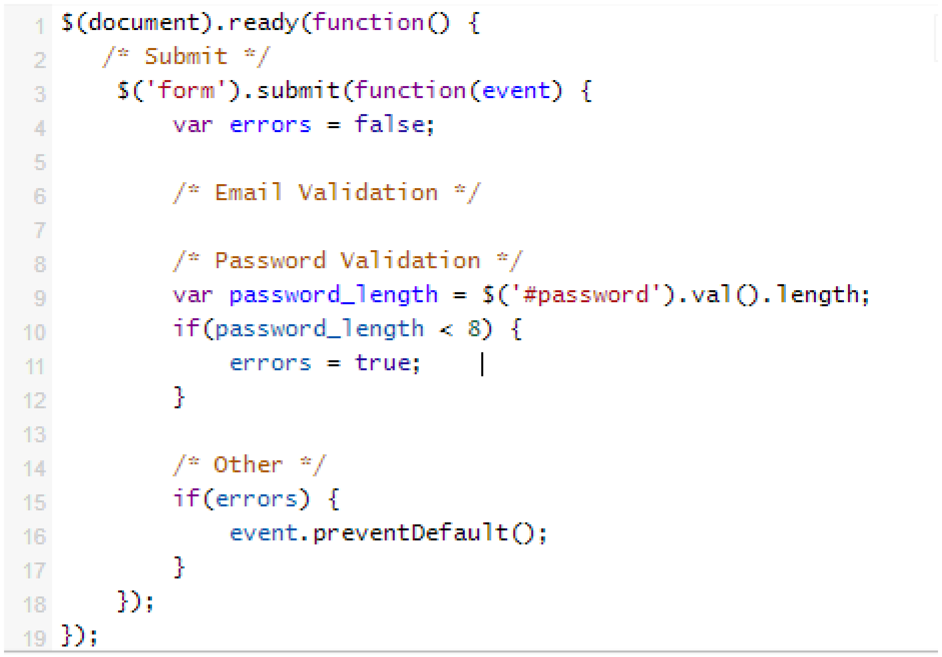
   Under /* Password Validation */ let's add some validation checks. First, let's get the value of our **#password** element, and then get its length. We know how to get value with **.val()**, and to get length you use **.length**. Let’s save this is a variable called **password_length**.



```
$(document).ready(function() {
   /* Submit */
    $('form').submit(function(event) {
        var errors = false;

        /* Email Validation */

        /* Password Validation */
        var password_length = $('#password').val().length;

        /* Other */
        if(errors) {
            event.preventDefault();
        }
    });
});
```

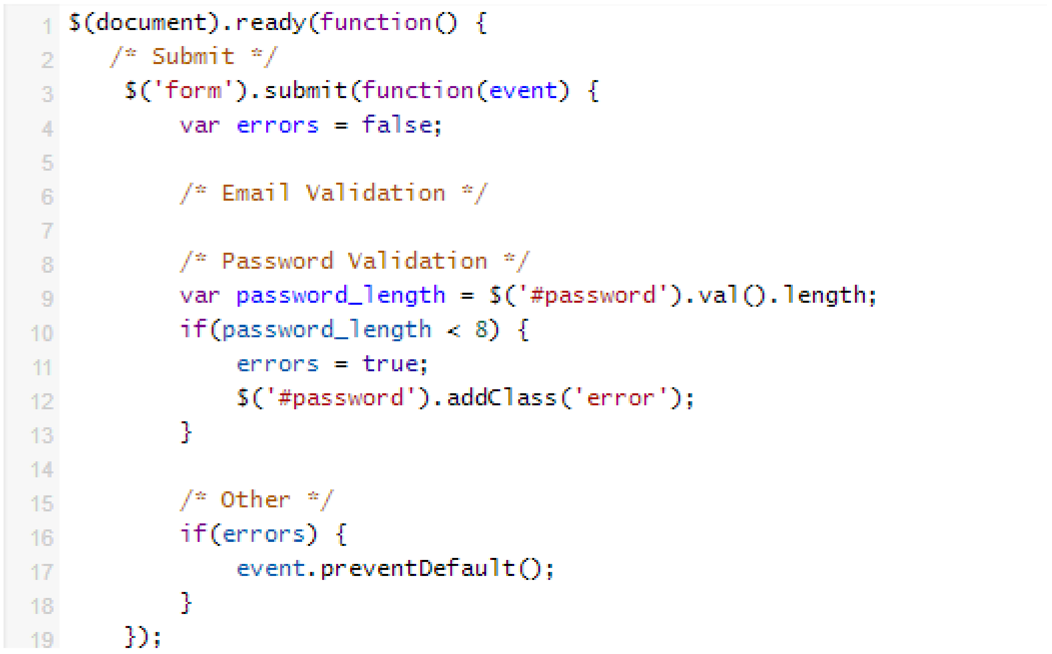
4. Next, let’s check if **password_length** is less than 8 characters long. If it is, let’s give an error to let the user know of this error.

   First, create an if statement for this case. Inside, also make it so that the variable **errors** is set to **true** (since we do not want to accept passwords less than length 8).

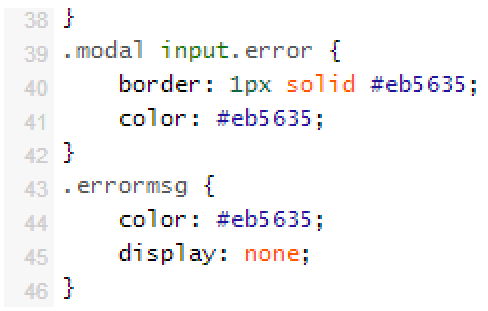


```
$(document).ready(function() {
   /* Submit */
    $('form').submit(function(event) {
        var errors = false;

        /* Email Validation */

        /* Password Validation */
        var password_length = $('#password').val().length;
        if(password_length < 8) {
            errors = true;
        }

        /* Other */
        if(errors) {
            event.preventDefault();
        }
    });
});
```

5. Next, inside our if statement, we want to add the class **error** to the password element. This is to give the password input box a red border and red text, to let the user know that there is something wrong with their password:



```
$(document).ready(function() {
   /* Submit */
    $('form').submit(function(event) {
        var errors = false;

        /* Email Validation */

        /* Password Validation */
        var password_length = $('#password').val().length;
        if(password_length < 8) {
            errors = true;
            $('#password').addClass('error');
        }

        /* Other */
        if(errors) {
            event.preventDefault();
        }
    });
```

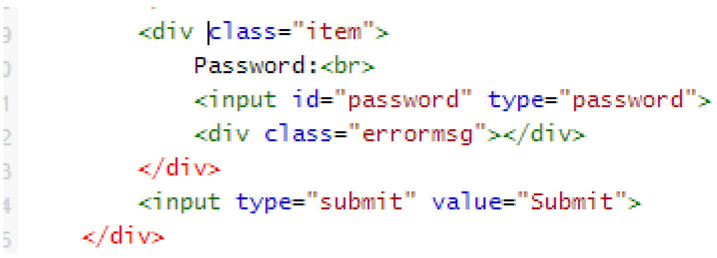


```
}
.modal input.error {
    border: 1px solid #eb5635;
    color: #eb5635;
}
.errormsg {
    color: #eb5635;
    display: none;
}
```

Notice the **.modal input.error** CSS style above.

Also notice above that we have a class **.errormsg** which is currently display none, but if our password is less than 8 characters, we want to display this message.

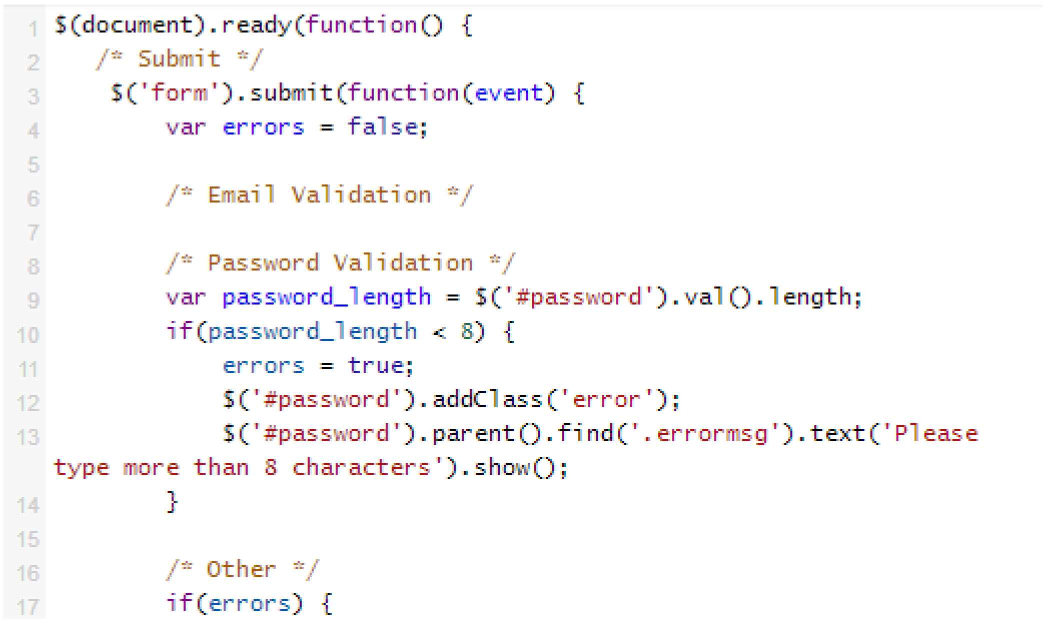
You can see this empty element **.errormsg** in our HTML:



```
    <div class="item">
        Password:<br>
        <input id="password" type="password">
        <div class="errormsg"></div>
    </div>
    <input type="submit" value="Submit">
</div>
```

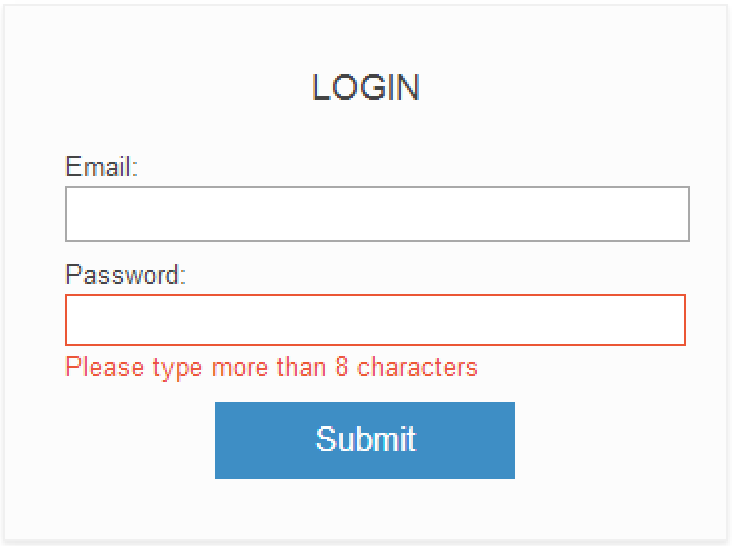
What we want to do is to select this element, make it appear with **.show()**, and put inside it some text telling the user what the error is (say “Please type at least 8 characters”).

Also notice that we can’t just select this error with **$(‘.errormsg’)** because Email also has the element. So, we can use DOM Traversal from **$(‘#password’)** to select this element.



```
$(document).ready(function() {
   /* Submit */
    $('form').submit(function(event) {
        var errors = false;

        /* Email Validation */

        /* Password Validation */
        var password_length = $('#password').val().length;
        if(password_length < 8) {
            errors = true;
            $('#password').addClass('error');
            $('#password').parent().find('.errormsg').text('Please type more than 8 characters').show();
        }

        /* Other */
        if(errors) {
```

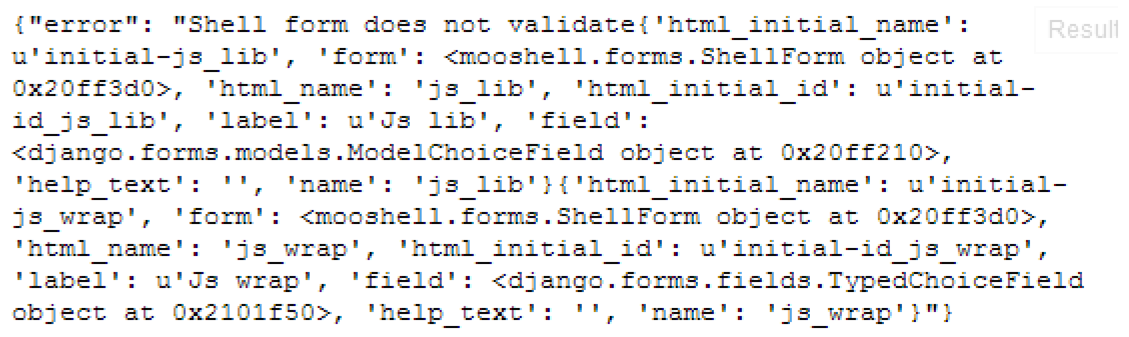
6. Now click **Run** and try clicking the Submit button! You should get an error:



Please type more than 8 characters

Nice! This means our validation for password is working.

7. Type some password with 8 or more characters and then click Submit. Again you will see:



```
{"error": "Shell form does not validate{'html_initial_name':
u'initial-js_lib', 'form': <mooshell.forms.ShellForm object at
0x20ff3d0>, 'html_name': 'js_lib', 'html_initial_id': u'initial-
id_js_lib', 'label': u'Js lib', 'field':
<django.forms.models.ModelChoiceField object at 0x20ff210>,
'help_text': '', 'name': 'js_lib'}{'html_initial_name': u'initial-
js_wrap', 'form': <mooshell.forms.ShellForm object at 0x20ff3d0>,
'html_name': 'js_wrap', 'html_initial_id': u'initial-id_js_wrap',
'label': u'Js wrap', 'field': <django.forms.fields.TypedChoiceField
object at 0x2101f50>, 'help_text': '', 'name': 'js_wrap'}"}
```

This means our form had no errors (other than the error above), at least on the front end, so all is good!

8. You can try form validations for Email too if you like, but we won't cover this in this activity. You can check if an email is formatted properly either in the backend, or in JavaScript using **Regex** (pattern matching, you can learn more online).

   Great job!

Congratulations! Let's review what we learned:

- **.submit()** events for forms
- Preventing a form from submitting if there are errors
- Adding error messages in the HTML
- Reviewed **.addClass()** if there is an error
- Reviewed DOM Traversal
- Basic Form Validation