Create, Read, Update, Delete (CRUD) with jQuery and AJAX
L10 – Manage Students

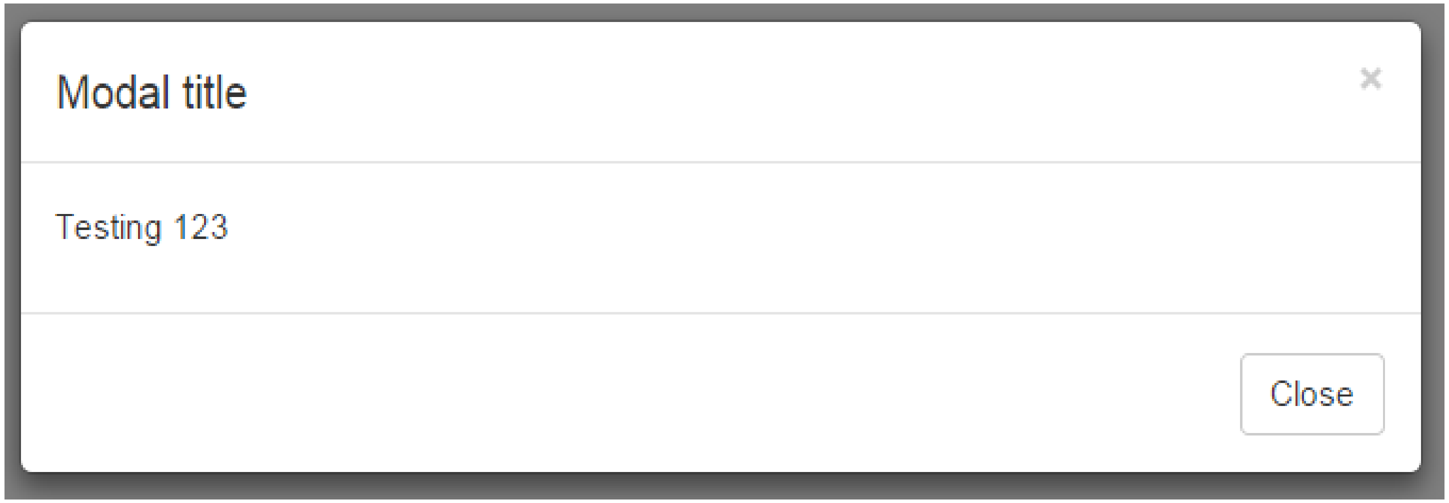
C273 – Advanced Web Application Development in PHP

Today's problem

- Create an application to perform Create, Read, Update and Delete functions in a single web page using AJAX and jQuery.

Bootstrap Modal

1. A modal is a child window that is layered over its parent window.
2. The purpose is to display content from a separate source that can have some interaction without leaving the parent window.



Testing 123

Bootstrap Modal

```
 <!-- Button trigger modal -->
       <button class="btn btn-primary btn-md" data-toggle="modal" data-target="#myModal">
Launch modal </button>
       <!-- Modal -->
       <div class="modal fade" id="myModal" role="dialog">
           <div class="modal-dialog">
               <div class="modal-content">
                   <div class="modal-header">
                       <h4 class="modal-title">Modal title</h4>
                   </div>
                   <div class="modal-body">
                       <p>Testing 123</p>
                   </div>
                   <div class="modal-footer">
                       <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
                   </div>
               </div><!-- /.modal-content -->
           </div><!-- /.modal-dialog -->
       </div><!-- /.modal -->
```

Header

Body

Footer

Dynamically created buttons

- Whenever the page reloads, an AJAX call is made to retrieve the list of modules and creates the table of records, Edit and Delete buttons.
- You can’t create a click callback function for .btnEdit using this:

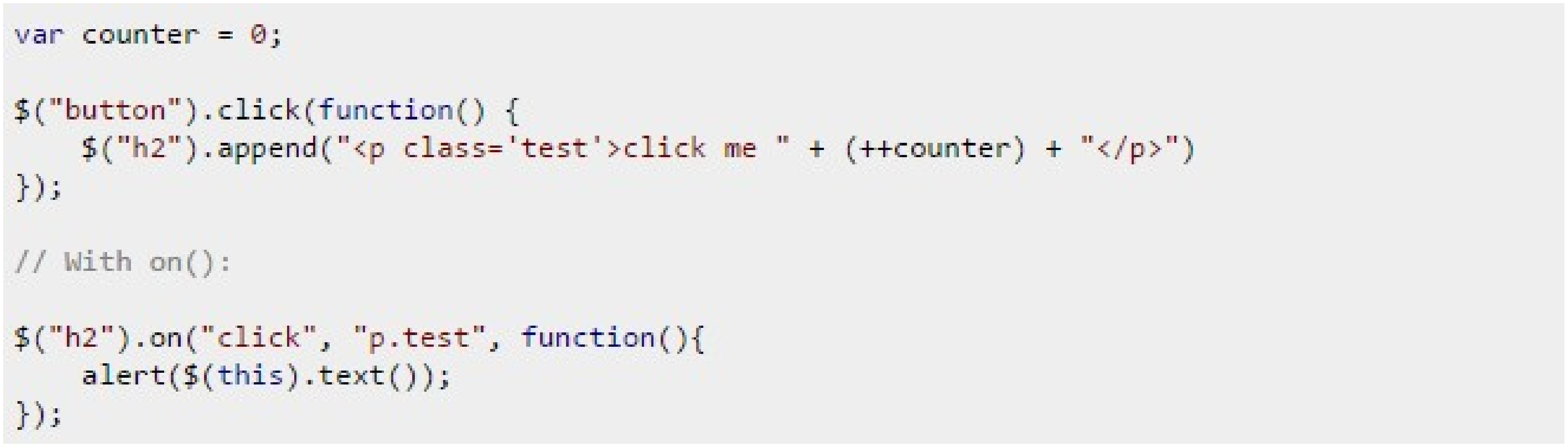
```
 $(“.btnEdit").click(function() {
});
```

- As the Edit buttons are generated dynamically
- You need to create a “delegated” binding using on() method:

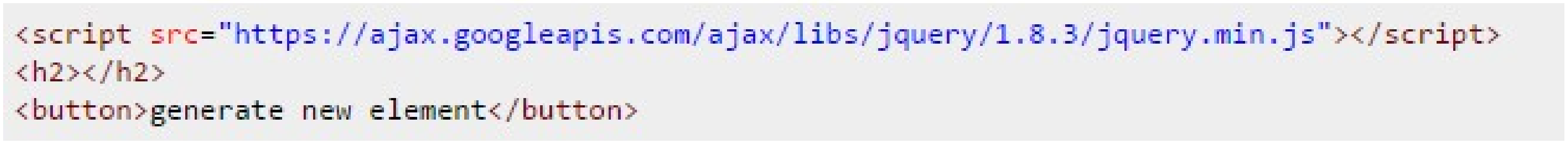
Delegated events

1. The click() binding you're using is called a "direct" binding which will only attach the handler to elements that *already exist*. It won't get bound to elements created in the future. To do that, you'll have to create a "delegated" binding by using **on()**.
2. Delegated events have the advantage that they can process events from descendant elements that are added to the document at a later time.
3. http://learn.jquery.com/events/event-delegation/

Delegated events



```
var counter = 0;

$("button").click(function() {
    $("h2").append("<p class='test'>click me " + (++counter) + "</p>")
});

// With on():

$("h2").on("click", "p.test", function(){
    alert($(this).text());
});
```



```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
<h2></h2>
<button>generate new element</button>
```

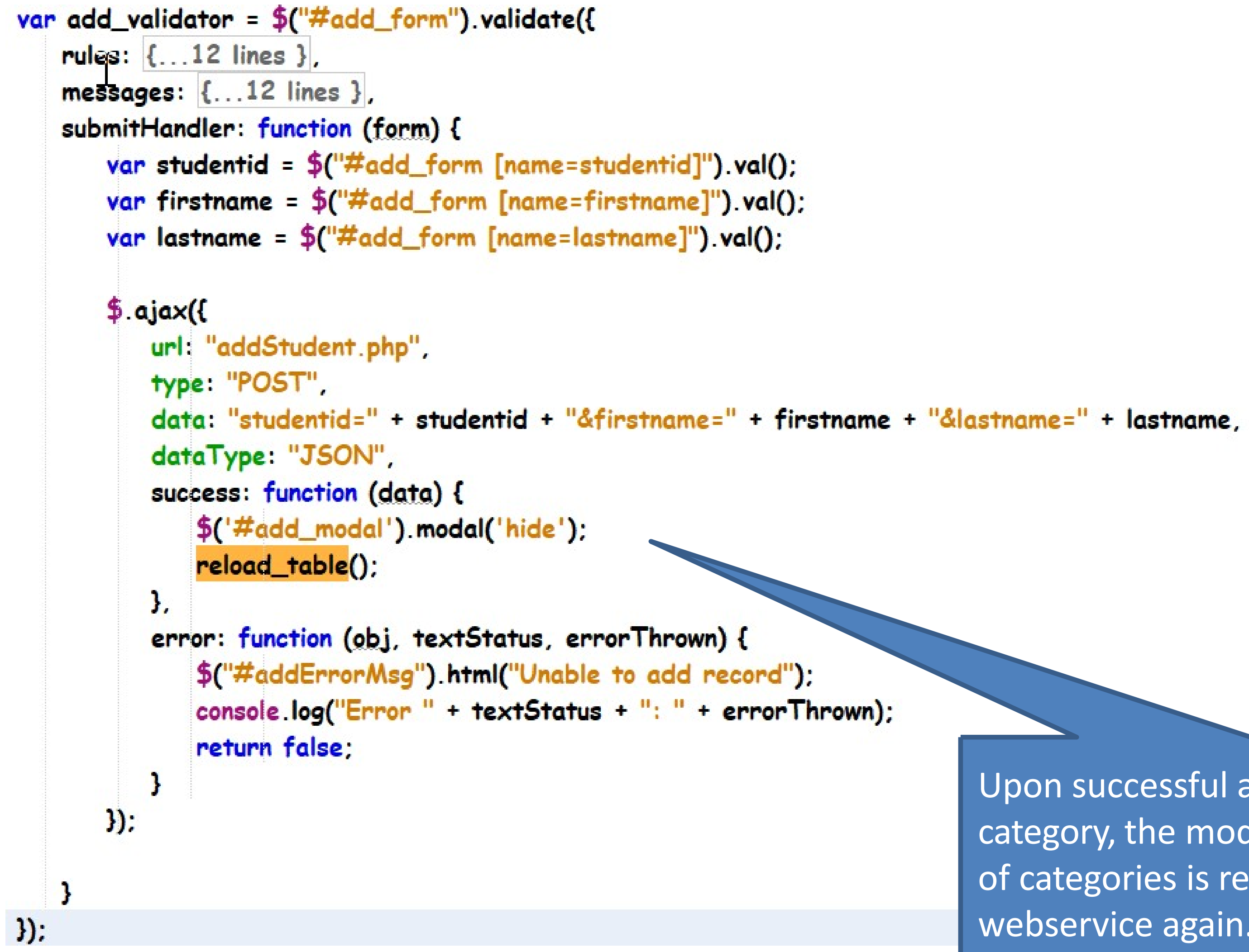
manageStudents.js

```
function reload_table() {

    $.ajax({
        type: "GET",
        url: "getStudents.php",
        cache: false,
        dataType: "JSON",
        success: function (response) {
            var message = "";
            for (i = 0; i < response.length; i++) {
                message += "<tr>"
                        + "<td>" + response[i].student_id + "</td>"
                        + "<td>" + response[i].first_name + "</td>"
                        + "<td>" + response[i].last_name + "</td>"
                        + "<td><button class='btnEdit btn btn-primary' value='" + response[i].student_id + <i class='fa fa-edit'></i>
Edit</button>&nbsp;&nbsp;"
                        + "<button class='btnDelete btn btn-danger' value='" + response[i].student_id + <i class='fa fa-trash'></i>
Delete</button></td>"

                        + "</tr>";
            }
            $("#defaultTable tbody").html(message);
        },
        error: function (obj, textStatus, errorThrown) {
            console.log("Error " + textStatus + ": " + errorThrown);
        }
    });
}
```

There are many Edit buttons, however "id" has to be unique for each button, therefore "class" is used

To identify which Edit button is clicked. We always use the primary key for the button value

manageStudents.js - continue



```
var add_validator = $("#add_form").validate({
    rules: {...12 lines },
    messages: {...12 lines },
    submitHandler: function (form) {
        var studentid = $("#add_form [name=studentid]").val();
        var firstname = $("#add_form [name=firstname]").val();
        var lastname = $("#add_form [name=lastname]").val();

        $.ajax({
            url: "addStudent.php",
            type: "POST",
            data: "studentid=" + studentid + "&firstname=" + firstname + "&lastname=" + lastname,
            dataType: "JSON",
            success: function (data) {
                $('#add_modal').modal('hide');
                reload_table();
            },
            error: function (obj, textStatus, errorThrown) {
                $("#addErrorMsg").html("Unable to add record");
                console.log("Error " + textStatus + ": " + errorThrown);
                return false;
            }
        });

    }
});
```

Upon successful adding or updating of the category, the modal is hidden and the list of categories is retrieved from the webservice again.

manageStudents.js

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

   reload_table();

 $('#add_modal').on('hidden.bs.modal', function () {
      $('#add_form')[0].reset();
      add_validator.destroy();
   });

   $("#btnAdd").click(function () {
      $('#add_form').modal('show');
   });
```

Reset the modal form input values once the modal is hidden to prevent the showing of previously entered values

Display the modal when the Add button is clicked

getStudentDetails.php JSON Message

PHP Code

```php
$studentID = $_GET['student_id'];
```

Test on web browser

http://localhost/C273Webservices/getStudentDetails.php?student_id=1111111

JSON Output

```
{
student_id: "1111111",
first_name: "Bob",
last_name: "Tan"
}
```

manageStudents.js - continue

```
$("#defaultTable").on("click", ".btnEdit", function () {
    var id = $(this).val();
    $.ajax({
      url: "getStudentDetails.php",
      data: "student_id=" + id,
      type: "GET",
      cache: false,
      dataType: "JSON",
      success: function (data) {
        $('#edit_form [name=studentid]').val(data.student_id);
        $('#edit_form [name=firstname]').val(data.first_name);
        $('#edit_form [name=lastname]').val(data.last_name);
        $('#edit_modal').modal('show');
      },
      error: function (obj, textStatus, errorThrown) {
        console.log("Error " + textStatus + ": " + errorThrown);
      }
    });
 });
```

You can't use: $(".btnEdit").click(function(){ as the Edit buttons are created dynamically. You need to use delegated binding

Retrieve Edit button's value

Retrieve JSON messages' keys

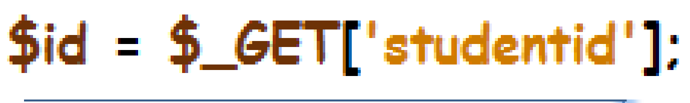
Set key values to modal form's input values.
Show modal

You need to specify $('#edit_form [name=lastname]') as the input field 'lastname' exists in #add_form as well

manageStudents.js - continue

```
$("#defaultTable").on("click", ".btnDelete", function () {
    var id = $(this).val();
    if (confirm('Are you sure you want to delete this student?')) {
      // ajax delete data to database
      $.ajax({
        url: "deleteStudent.php",
        data: "studentid=" + id,
        type: "GET",
        dataType: "JSON",
        success: function (data) {
          reload_table();
        },
        error: function (obj, textStatus, errorThrown) {
          console.log("Error " + textStatus + ": " + errorThrown);
        }
      });
    }
  });
```

Same as deleteStudent.php



```
$id = $_GET['studentid'];
```

What you learnt today

1. Apply Bootstrap components, Modal and FontAwesome icons, on web pages
2. Perform form submission using AJAX and jQuery via the POST method
3. Explain the difference between direct event binding and delegated event binding
4. Apply event delegation using the on() method
5. Apply AJAX and jQuery to build a datagrid that dynamically add, edit and delete rows in a single web page.