PHP AJAX

Level 1 - Passing Information

1. Create a simple form on a web page (without a form tag). The form should act as a registration form. Ask the user for their first name and last name. Then, use JQuery AJAX to send the information to a registration processing php page. This page should take the first initial of the first name and the first five characters of the last name and concatenate them. Finally, add a random two digit number 00 - 99 to the end of the name and return it to the calling page. On the calling page, simply alert their new username.
2. Create a generic order form. The order form should have 5 items listed and each item should have a quantity area. On submit, the information should be sent to a php page. There, the first items should cost $.75 each, the second type should be $1.00 each, the third item should be $1.35, the fourth should be $2.00 and the fifth should be $5.43. Have the php page calculate the total and return it to the original page. Then, have the calling page alert the total.
3. Create a page with three buttons. Clicking each should get a value from a hidden input. Send the value to a php page. Based on the value coming to the page (1, 2 or 3), return a paragraph of text to the calling page and populate a div with the returned content.
4. Using the code from level 1, problem 2, When the total is returned, show it in an uneditable text input. Then, use it to calculate and display the sales tax (6%) and finally show the final total.

Level 2 - Incorporating MySQL (NO JQuery DOM necessary)

1. Using the Blog table in the Content database from the MySQL exercises, write a php page that shows a drop down list of all of the authors. When a user selects an author, use AJAX to call a php page that queries the database and returns the number of blog entries that author has entered. Show the number in an alert on the original page.
2. Using the Auditions database from the MySQL exercises, write a php page that shows a drop down list of audition numbers, three inputs - one for the acting score, one for the music score and one for the dance score, and a blue (Bootstrap info) submit button. If that audition number exists in the scores table, populate the appropriate scores. Otherwise, leave the inputs blank. When the user submits the scores, use AJAX to either add to or update the scores table. If the operation is successful, turn the original button green (Bootstrap success), otherwise, turn the original button red (Bootstrap danger).
3. Using the Conferences database from the MySQL exercises, write a php page that has an input for an email and a “Get Schedule” button. When the user clicks the button, use AJAX to call a php page that will query the database and get the number of conferences the person has. Return to the calling page, the type of user they are (Teacher or Parent) and the number of conferences they have scheduled. You can either push to a pre-exisitng div OR simply alert the information.

Level 3 - Incorporating MySQL (JQuery DOM required)

1. Use the blogs table to create a page with all of the blogs on it. When a user hits the page, there should be a sort option at the top of the page (title, author, date). Use ajax to query the database and rebuild the page with the appropriate sort.
2. Using the conferences database, create an admin page that shows all conferences in order of date/time. At the top of the page should be a search bar that allows an admin to search for a particular user. Searching for a user should involve typing in their last name. Submitting the search should regenerate the list of conferences for just that user (regardless of whether they are a parent or teacher). The regeneration should be accomplished with JQuery and without reloading the page. There should also be a button to clear the search. For an extra challenge, create a way to choose between users if there are multiple users that have the same name.
3. Use the auditions database and create a page that shows the auditionees. Start by altering the table to have an audition number for each auditionee. Girls with last names A-L should start at 100, girls with

last names M-Z should start at 200 and boys should start at 300. Also, alter the table to have a gender for each auditionee. On the page, each auditionee should be shown with their audition number, last name, first name and gender. Display the auditionees by their audition number. Additionally, if they have scores, display each of the scores and the total of the scores. On the page should be a quick registration form. The quick reg should ask just for a students first, last, gender and email. When the form is submitted, add the auditionee to the database, assign them the next appropriate audition number, and display them on the page in the appropriate position without reloading the page.