Great work! We covered a lot in this lesson. Let’s review:

* Performing back-end *form validations* on the data submitted is an essential step to protect our website and its users.
* Using the POST method attribute in an HTML form gives our PHP script access to data submitted within the superglobal associative array: $\_POST.
* We modify our HTML and PHP so that when input is deemed invalid, meaningful feedback is shown to the user.
* If we plan on displaying user input, we need to first *sanitize* it. We can use methods like trim() and htmlspecialchars() for basic sanitization.
* We can use filter\_var() with a filter to sanitize common input types.
* We can also use filter\_var() with a filter to perform validations on common input types.
* We’ll often want to perform custom validations.
* The preg\_match() function compares checks if a given string matches a regular expression.
* Since regular expression comparisons can consume a lot of computing power, we’ll want to check the length of inputs before performing regular expression checks.
* It’s common to perform validations by comparing user input to back-end data
* Before storing user input in our back-end, we’ll sanitize it for both safety and consistent formatting
* If a user’s form submission has been accepted, we can reroute them to a different page.

Data validation and sanitization is an extremely important part of web development. In this lesson, we’ve covered some of the basic theory and techniques. When developing for production, you’ll need to further research and understands the needs of the specific sites or applications as well as the tools available with the specific databases or frameworks in use.

As your validations get more and more complex, you should also [practice *modularity*](https://www.codecademy.com/courses/learn-php/lessons/logical-operators-and-compound-conditions/exercises/multi-file-programs-include?action=resume_content_item) and separate your validation logic from your display logic.