Input Validation, Sanitization and PDO in Forms on the Server Side

RRC Polytech

Full Stack Web Development

Winnipeg, MB Canada

What is Validation and Sanitization Mean

- Data Validation: Checking Foreign Input
  - with JavaScript we've checked to make sure that the postal code is the right format that's checking for an input
- Data Sanitization: Cleaning Foreign Input
  - Never trust foreign data

What Is Foreign Data Sources

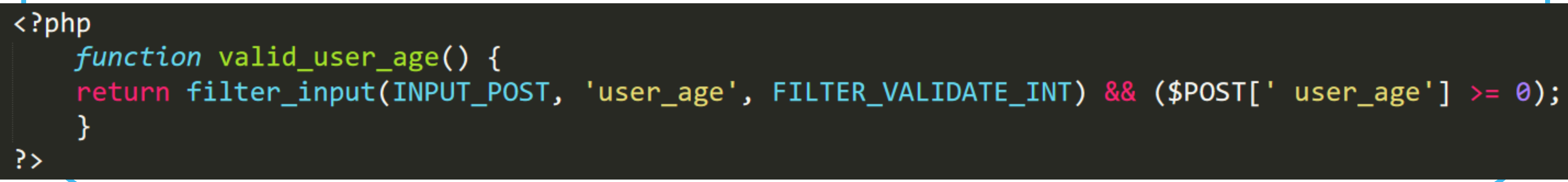
- Foreign data sources
  - $_GET, $_POST and … superglobals
  - data can be passed through session cookie values
  - 3rd party API requests (getting data from external sources)
  - internal data integration systems (get data from sources within your organization)

Validation Vs. Sanitization

- Validation is verifying that data conforms to the rules you set for a particular input field. For example, when we ask for a user's age, we expect a positive number in return.
- Sanitization is filtering data to remove corrupt or harmful information. For example, to block SQL-Injection or Cross-Site Scripting (XSS) attacks.

Validations With Filter input()

- Simple validations are often required to ensure that the input is present and reasonable. We might also need to ensure that our data matches the column type expectations of the database table where it will be stored.
- For example, ensuring that a user's age was provided and that it is a positive whole number.



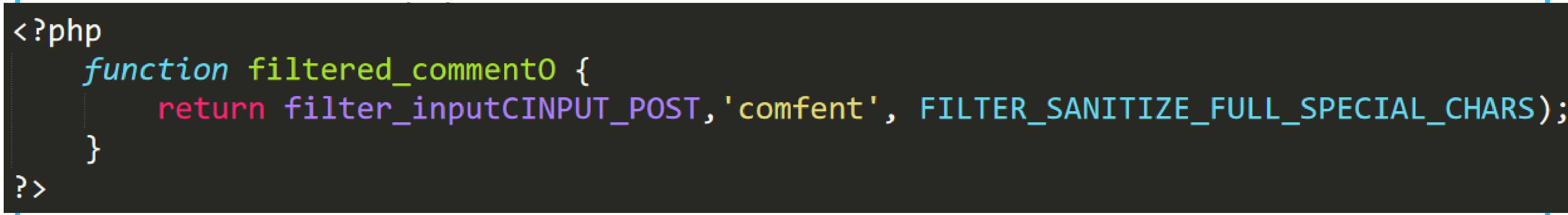
```php
<?php
    function valid_user_age() {
    return filter_input(INPUT_POST, 'user_age', FILTER_VALIDATE_INT) && ($POST[' user_age'] >= 0);
    }
?>
```

Validation Test Parameters

- input filter function should be used and it takes normally 3 parameters (for more information use php.net)
- The first option is where is the data coming from and it's in this case it's coming from the Super global and that's represented as INPUT_POST (so it's coming from the GET would be input_get)
- The second parameter is what key are we going to use to filter (here we've got an example where we're using the user_age)
- What filter do we want to apply to now this is where you will want to go to php.net because <u>there are a lot of these filter options</u> that we have
- we're appending on our own validations just to make sure that it's greater than or equal to 0 (>=0)
- there is also a filter validate positive integer so you wouldn't even need this appended code here just say filter value positive
- we've got these inner function here so return true if it passes this validation test.

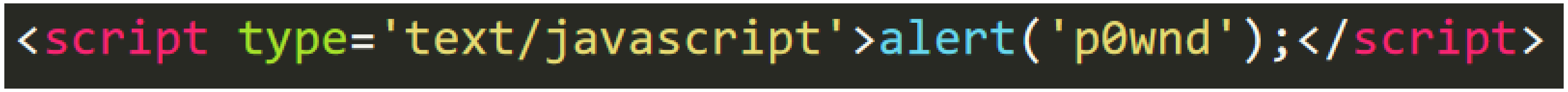
Sanitization With Filter input

- Sanitization can be used to prevent HTML injection by converting certain characters to



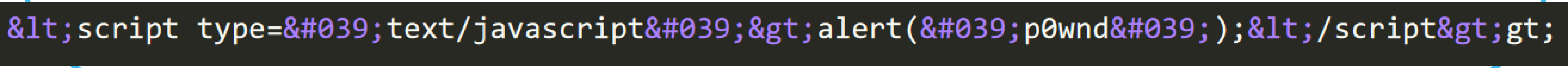
```
<?php
    function filtered_commentO {
        return filter_inputCINPUT_POST,'comfent', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
    }
?>
```

- The malicious comment:



```
<script type='text/javascript'>alert('p0wnd');</script>
```

- Becomes:



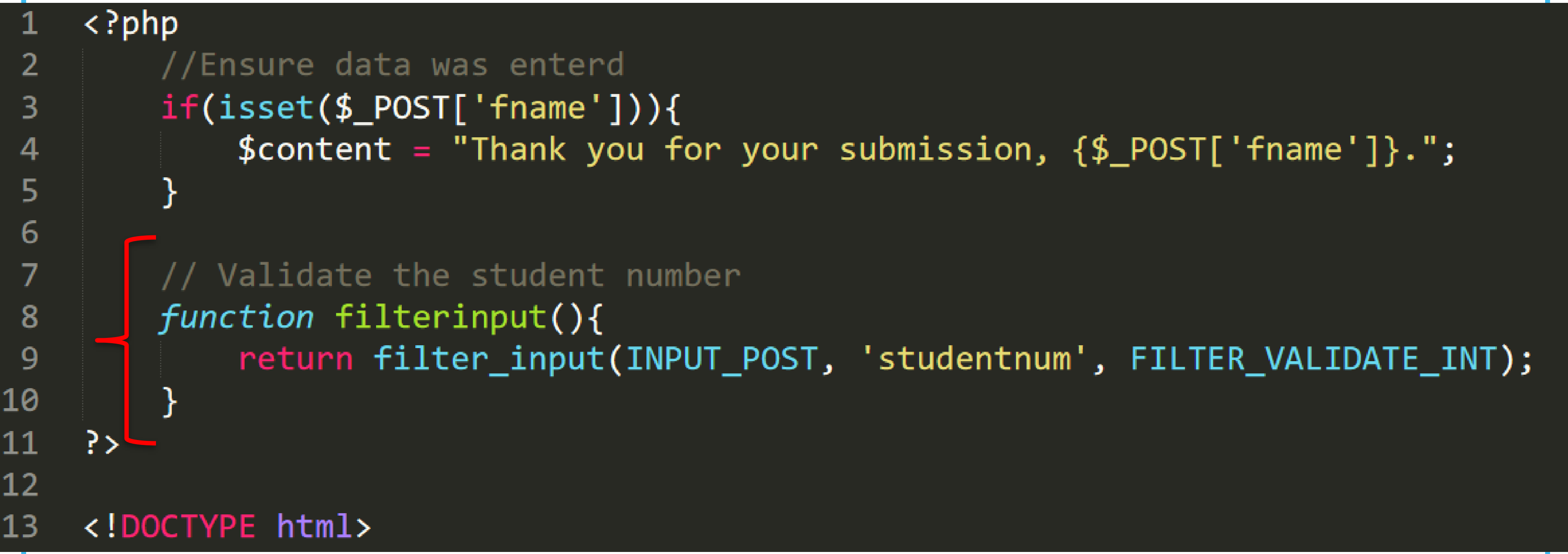
```
&lt;script type=&#039;text/javascript&#039;&gt;alert(&#039;p0wnd&#039;);&lt;/script&gt;gt;
```

Sanitization

- How Sanitization works?
  - It removes those special characters out so that's

RRC POLYTECH

In the thankyou.php file, Add Validation for Student Number

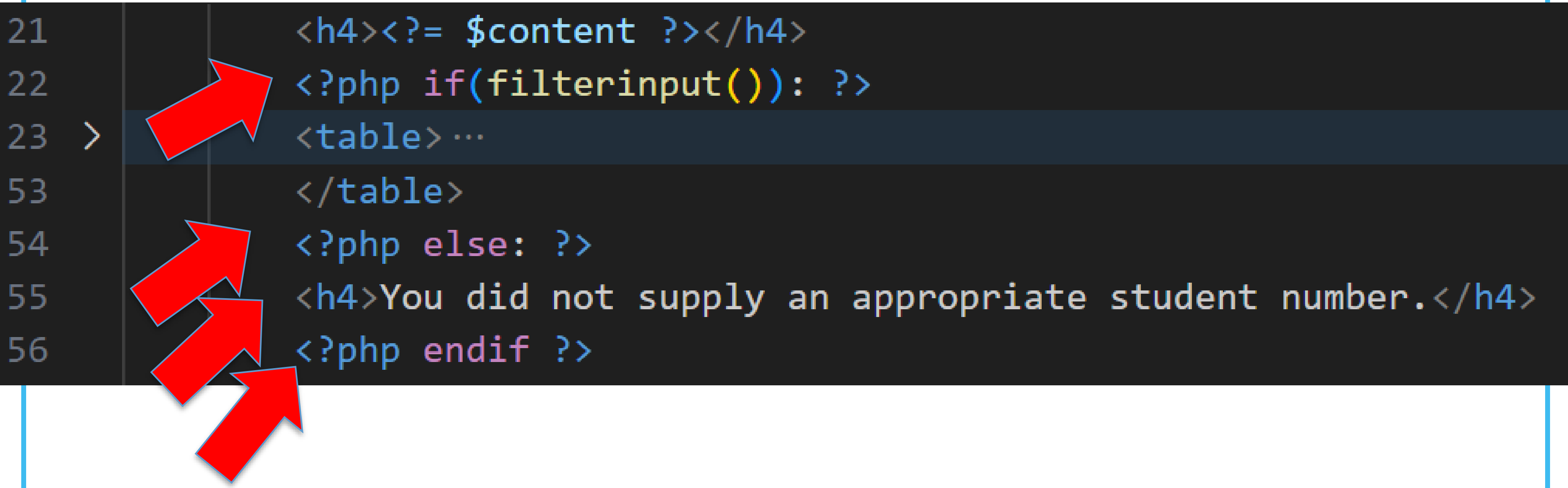


```php
<?php
    //Ensure data was enterd
    if(isset($_POST['fname'])){
        $content = "Thank you for your submission, {$_POST['fname']}.";
    }

    // Validate the student number
    function filterinput(){
        return filter_input(INPUT_POST, 'studentnum', FILTER_VALIDATE_INT);
    }
?>

<!DOCTYPE html>
```

Parameters

- The first option is where is the data coming from and it's in this case it's coming from the Super global and that's represented as INPUT_POST (so it's coming from the GET would be input_get)
- The second parameter is what key are we going to use to filter (here we've got an example where we're using the user_age)
- What filter do we want to apply

In the thankyou.php file, Add Validation for Student Number

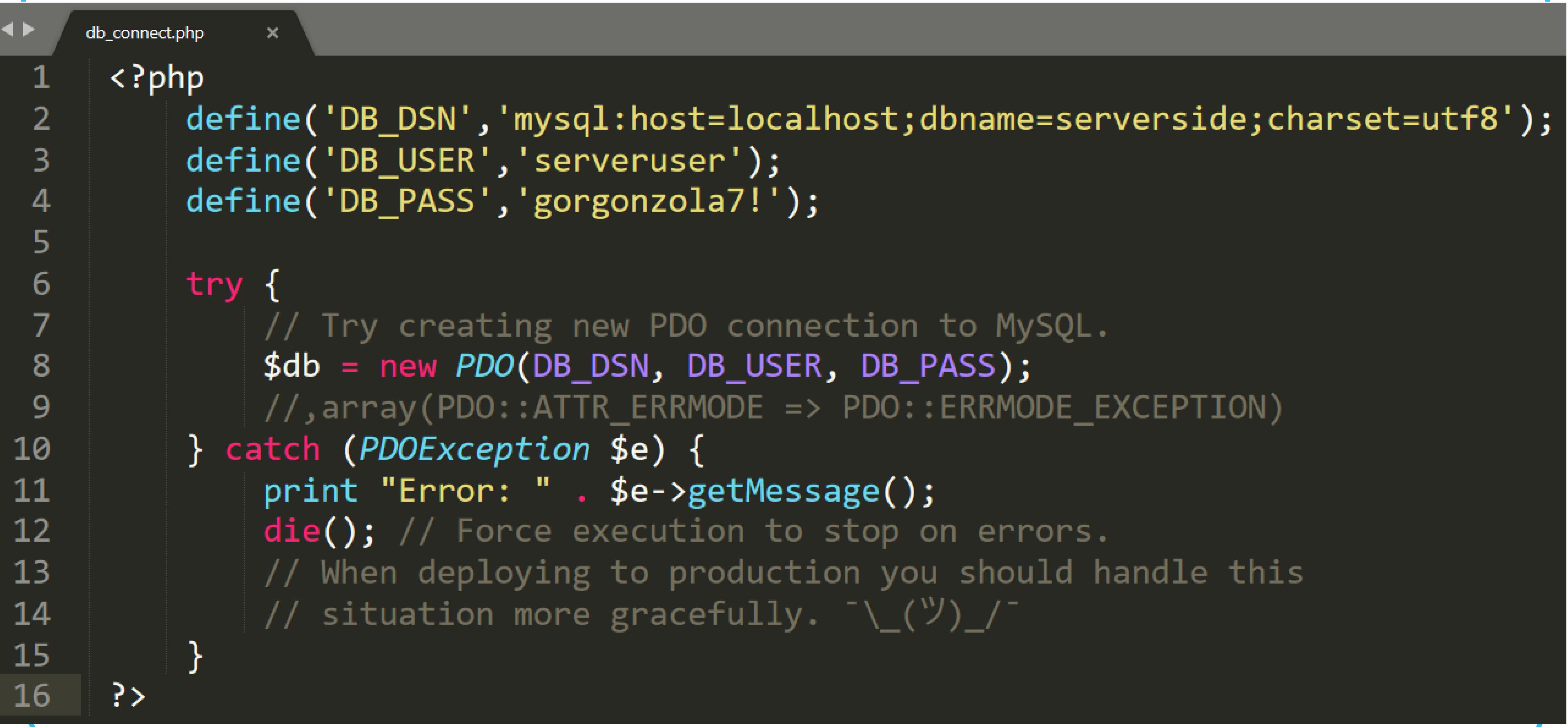


```
        <h4><?= $content ?></h4>
        <?php if(filterinput()): ?>
        <table>···
        </table>
        <?php else: ?>
        <h4>You did not supply an appropriate student number.</h4>
        <?php endif ?>
```

RRC POLYTECH

PDO
(PHP Data Objects)

"db_connect" File Shows How do we Connect php Code to Our Database



db_connect.php

```php
<?php
	define('DB_DSN','mysql:host=localhost;dbname=serverside;charset=utf8');
	define('DB_USER','serveruser');
	define('DB_PASS','gorgonzola7!');

	try {
		// Try creating new PDO connection to MySQL.
		$db = new PDO(DB_DSN, DB_USER, DB_PASS);
		//,array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION)
	} catch (PDOException $e) {
		print "Error: " . $e->getMessage();
		die(); // Force execution to stop on errors.
		// When deploying to production you should handle this
		// situation more gracefully. ¯\_(ツ)_/¯
	}
?>
```

Info

- define my connection with constants (line 2 to 4)
  1. DSN: Data Source Name
     - Define a mysql database
     - With PDO we connect to different kinds od databases; in this case we use mysql
     - Our host is localhost
     - Database name is “serverside”
     - Charset=utf8 -> Unicode transformation format 8 bit is used for encoding evry character => the php website can handle and deslay text in multi language
  2. Database username: for everybody in the class we have the same username
  3. Database password: for everybody in the class we have the same password

Info

- Line 8: PDO (php data object)
- We createa new php data object and we call it $db
- When you enable line 9, you enable to see errors on the screen
- If we are not successful to create database connection, we print the error message; and we kill the app.

PHP Data Objects (PDO)

- The PHP Data Objects extension (PDO) allows you to access the functionality provided by a SQL server in an object-oriented manner.

- Provided by the PDO extension is the MySQL Driver which represents a connection between PHP and a MySQL database. By changing the PDO Driver you can switch between different types of database servers.

RRC POLYTECH

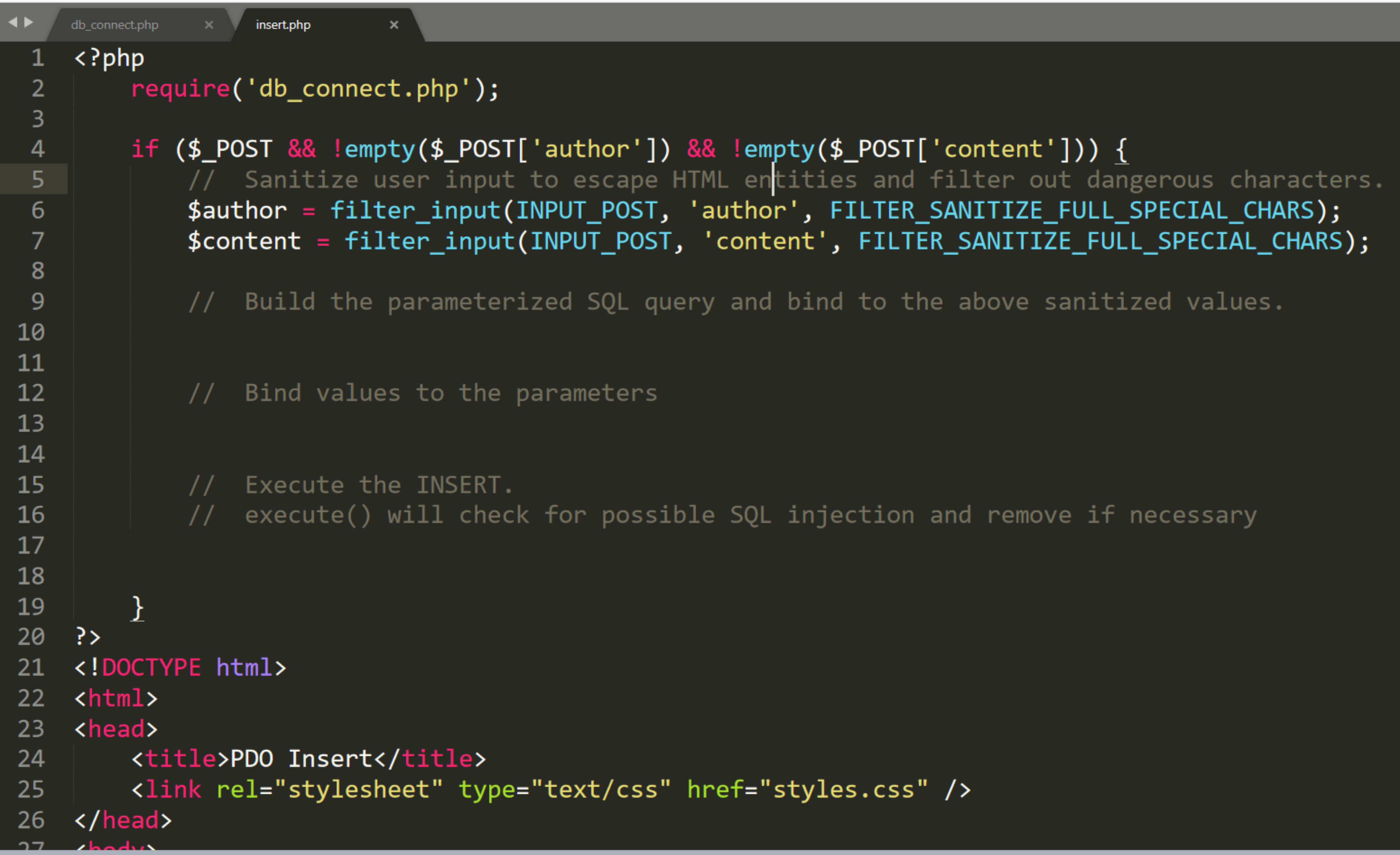
Remember

- In PHP object methods are called using the stabby operator -> as follows:

```
$favourite_noun->the_best_method_in_the_world($coolest_parameter_evar);
```

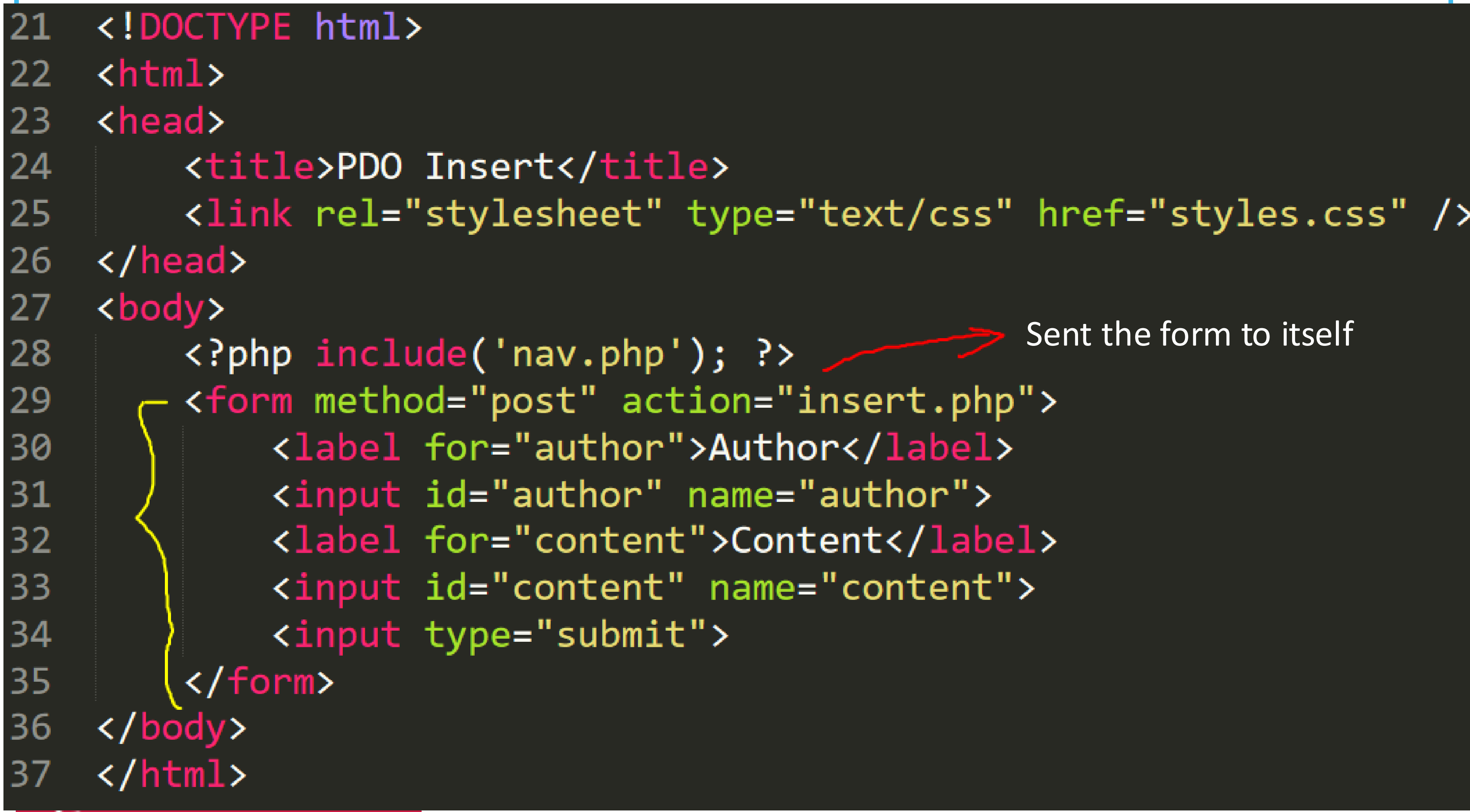
- Assuming we’ve already instantiated an object called $favourite_noun.

Open "insert" File



```
<?php
    require('db_connect.php');

    if ($_POST && !empty($_POST['author']) && !empty($_POST['content'])) {
        //  Sanitize user input to escape HTML entities and filter out dangerous characters.
        $author = filter_input(INPUT_POST, 'author', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
        $content = filter_input(INPUT_POST, 'content', FILTER_SANITIZE_FULL_SPECIAL_CHARS);

        //  Build the parameterized SQL query and bind to the above sanitized values.


        //  Bind values to the parameters


        //  Execute the INSERT.
        //  execute() will check for possible SQL injection and remove if necessary


    }
?>
<!DOCTYPE html>
<html>
<head>
    <title>PDO Insert</title>
    <link rel="stylesheet" type="text/css" href="styles.css" />
</head>
```

HTML Section of “insert” File

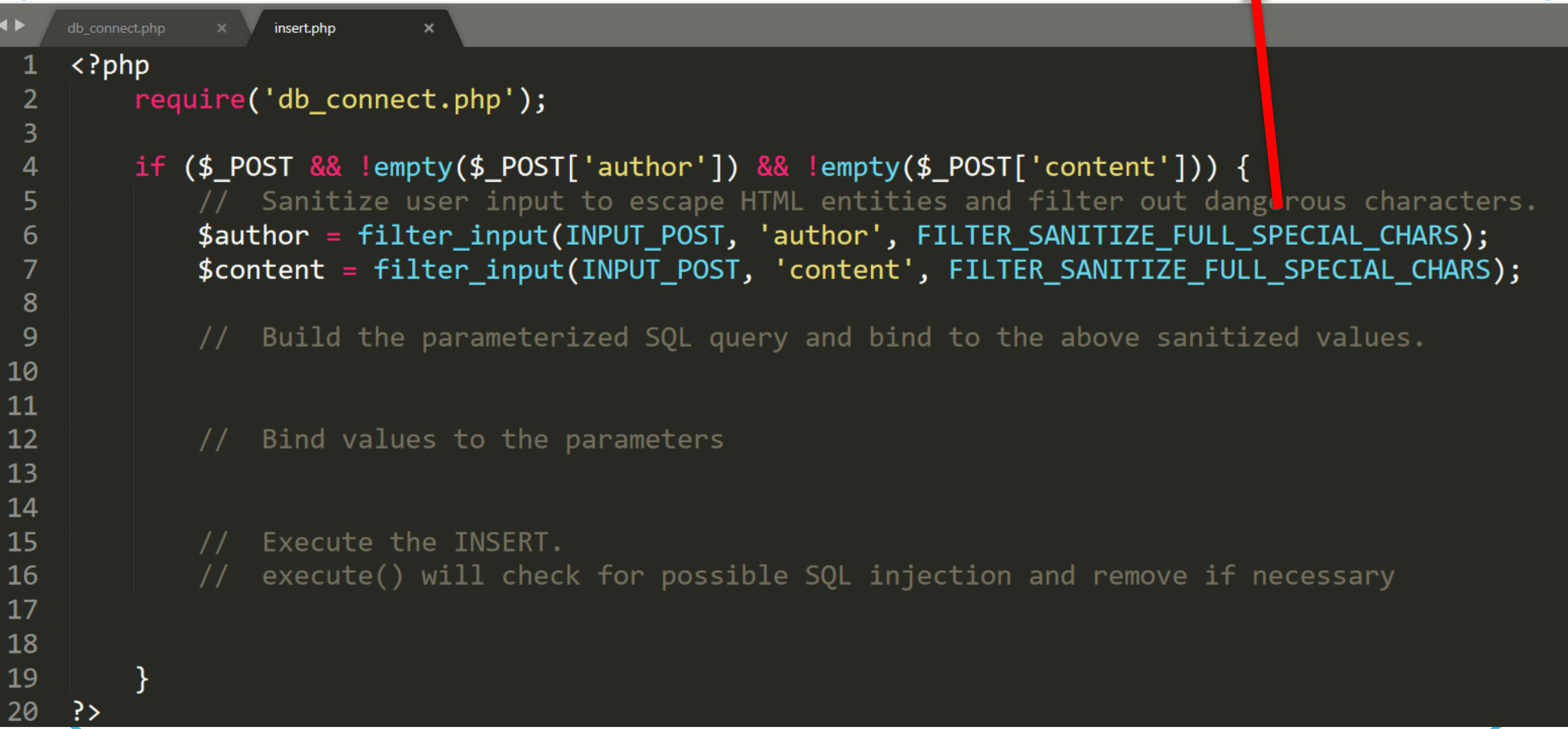


```
<!DOCTYPE html>
<html>
<head>
    <title>PDO Insert</title>
    <link rel="stylesheet" type="text/css" href="styles.css" />
</head>
<body>
    <?php include('nav.php'); ?>
    <form method="post" action="insert.php">
        <label for="author">Author</label>
        <input id="author" name="author">
        <label for="content">Content</label>
        <input id="content" name="content">
        <input type="submit">
    </form>
</body>
</html>
```

Sent the form to itself

php Section of "insert" File

Mitigate SQLI, javascript injection and XSS attacks



```php
<?php
    require('db_connect.php');

    if ($_POST && !empty($_POST['author']) && !empty($_POST['content'])) {
        //  Sanitize user input to escape HTML entities and filter out dangerous characters.
        $author = filter_input(INPUT_POST, 'author', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
        $content = filter_input(INPUT_POST, 'content', FILTER_SANITIZE_FULL_SPECIAL_CHARS);

        //  Build the parameterized SQL query and bind to the above sanitized values.


        //  Bind values to the parameters


        //  Execute the INSERT.
        //  execute() will check for possible SQL injection and remove if necessary


    }
?>
```

Info: Filtering

- The **filter input function** runs on two inputs
  - FILTER_SANITIZE_FULL_SPECIAL_CHARS

- The **filter sanitize full special characters** is used to filter **SQL injection**, **JavaScript injection** or **cross site script injection**.
  - htmlspecialchar()

Info: PDO

- using **PDO**, you don't just execute a write SQL statement
- they build what's called **parameterized SQL queries** prepared statements is another term
- you may have heard in database
- what PDO does?
  - is actually give us a real security advantage so **SQL injections can't actually happen** in this case of using PDO
  - Example: the little Bobby tables example uses the parameter would just be enclosed (enveloped) in quotes so that would **mitigate any kind of SQL injection** using this **parameterized SQL format**

RRC POLYTECH

Thank you