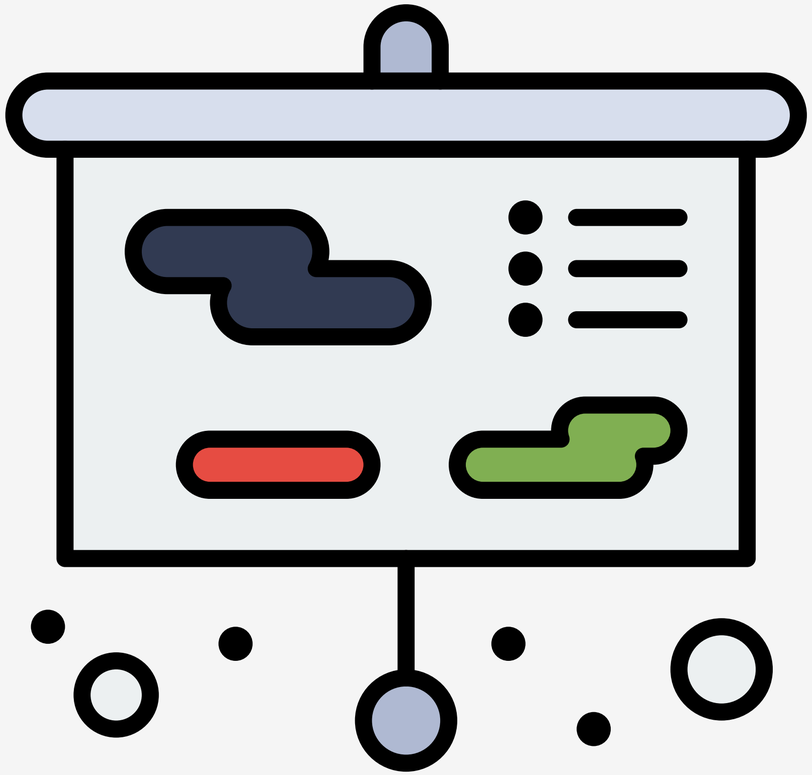
SERVER-SIDE
WEB DEVELOPMENT

Lecture 6

TODAY'S TOPICS

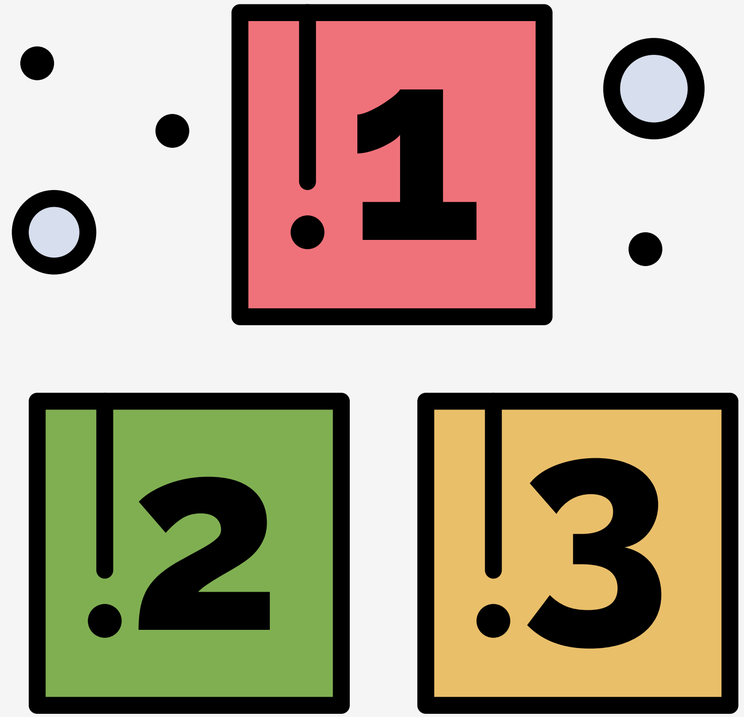


- PHP Data Objects
- Project: Seussology

QUESTIONS

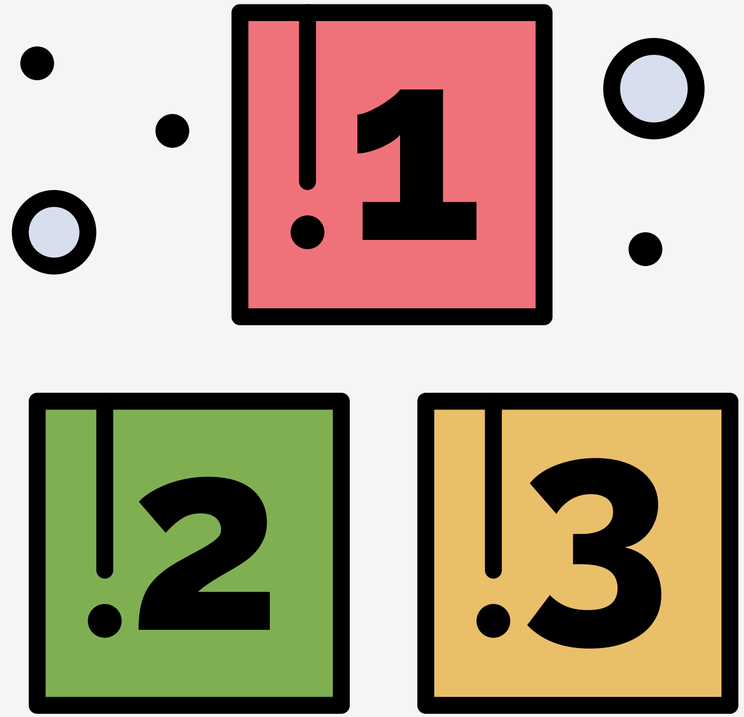
PHP DATA OBJECTS

PHP DATA OBJECTS (PDO)



- PDO is a PHP interface for accessing and manipulating databases
- PDO is database independent
- PDO using Object Oriented programming

CONNECTING WITH PDO

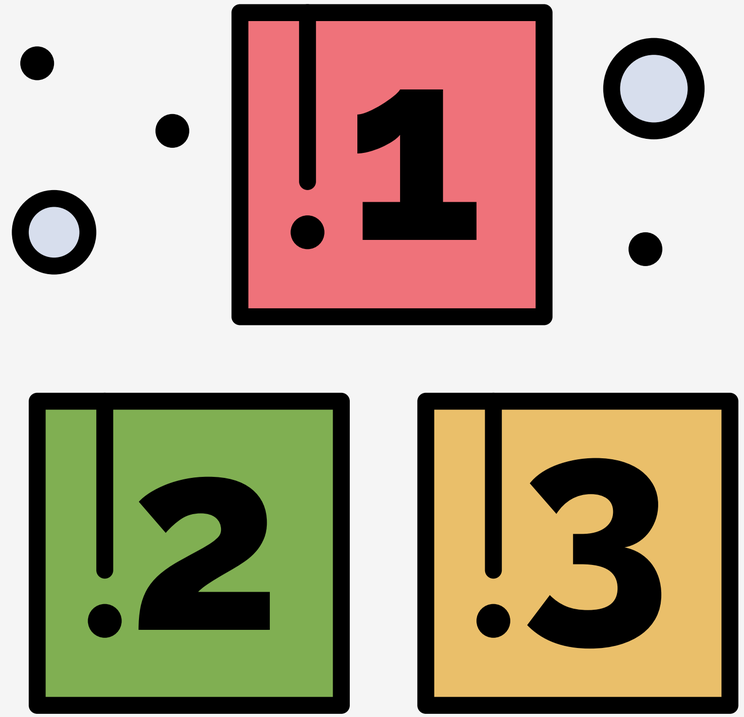


- Connecting to a database with PDO depends on the DBMS
- All examples will be for MySQL
- Three parts are required:
  - Database Source Name (DSN)
  - Username
  - Password
- A try / catch block should be used

CONNECTING WITH PDO

```php
<?php
  $dsn = 'mysql:host=localhost;dbname=movies';
  $user = 'root';
  $pass = 'root';

  try {
    $db = new PDO($dsn, $user, $pass);
  } catch (PDOException $e) {
    print $e->getMessage() . "<br/>";
    die();
  }
```

SELECT WITH PDO



- The `query()` method can be used to execute `SELECT` statements
- The `query()` method will return a PDOStatement object
- The `fetch()` or `fetchAll()` method can be used to get the results
- The `fetch()` method gets one result at a time, typically used with a `while` loop
- The `fetchAll()` method get all the results as an array, typically used with a `foreach` loop

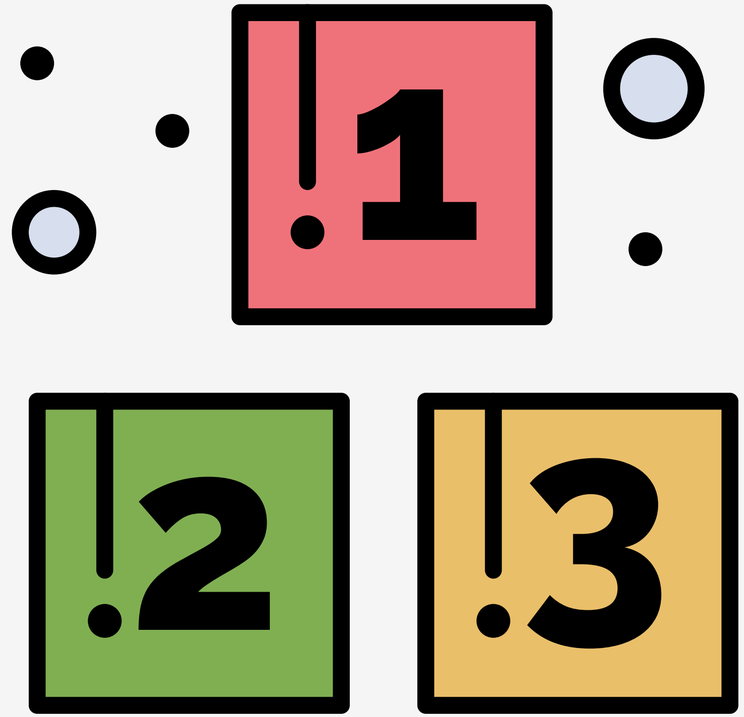
SELECT WITH PDO

```php
<?php
  require_once "db.php";
  $sql = "SELECT * FROM movies";
  $result = $db->query($sql);
?>

<ul>
<?php while ($movie = $result->fetch()) : ?>
  <li><?php echo $movie['movie_title']; ?></li>
<?php endwhile; ?>
</ul>
```

SELECT WITH PDO

```php
<?php
  require_once "db.php";
  $sql = "SELECT * FROM movies";
  $result = $db->query($sql);
  $movies = $result->fetchAll();
?>

<ul>
<?php foreach ($movies as $movie) : ?>
  <li><?php echo $movie['movie_title']; ?></li>
<?php endforeach; ?>
</ul>
```

NON-SELECT WITH PDO



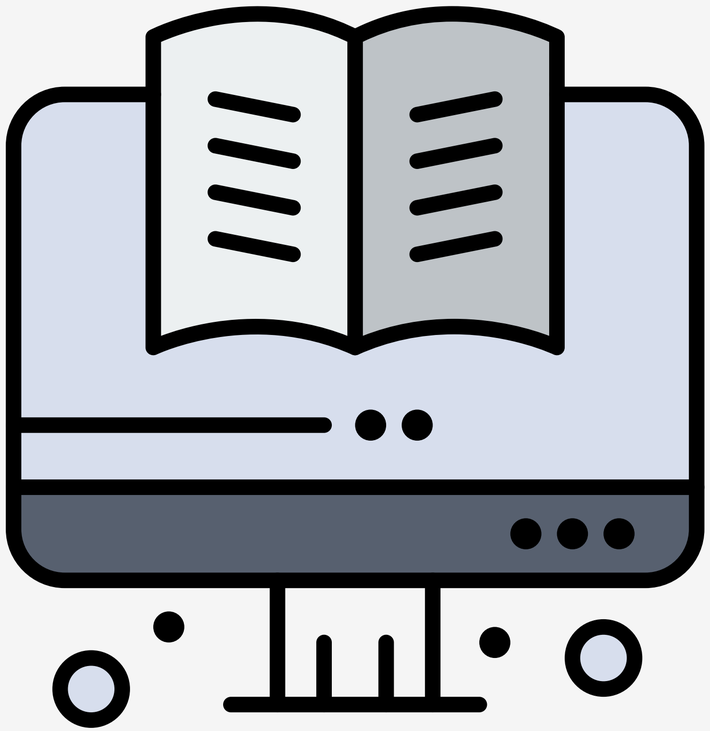
- The `exec()` method can be used to execute `NON-SELECT` statements
- The `exec()` method will return the number of rows affected

NON-SELECT WITH PDO

```php
<?php
  require_once "db.php";
  $sql = "INSERT INTO movies (`movie_title`)
          VALUES ('Pulp Fiction')";
  $result = $db->exec($sql);
  echo $result // 1
```

ERROR HANDLING

ERROR HANDLING



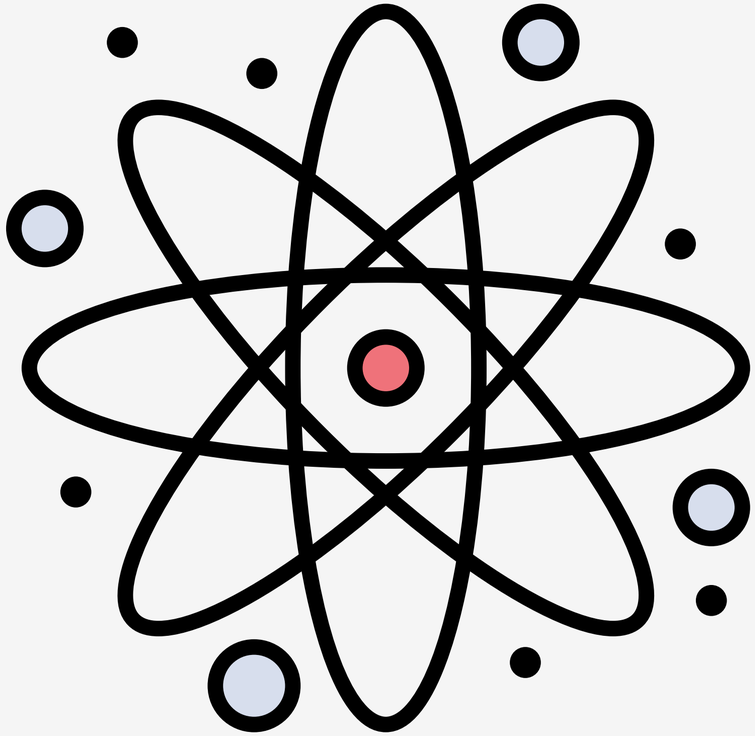
- When an error is in the SQL, PHP will not be able to tell
- The `errorInfo()` method returns an array containing error information
- The third item in the array is the human-readable explanation of the error

ERROR HANDLING

```php
<?php
  require_once "db.php";
  $sql = "SELECT * FROM movis";
  $result = $db->query($sql);
  $errorInfo = $db->errorInfo();

  echo $errorInfo[2]; // no such table: movis
```

PREPARED STATEMENTS

PREPARED STATEMENTS

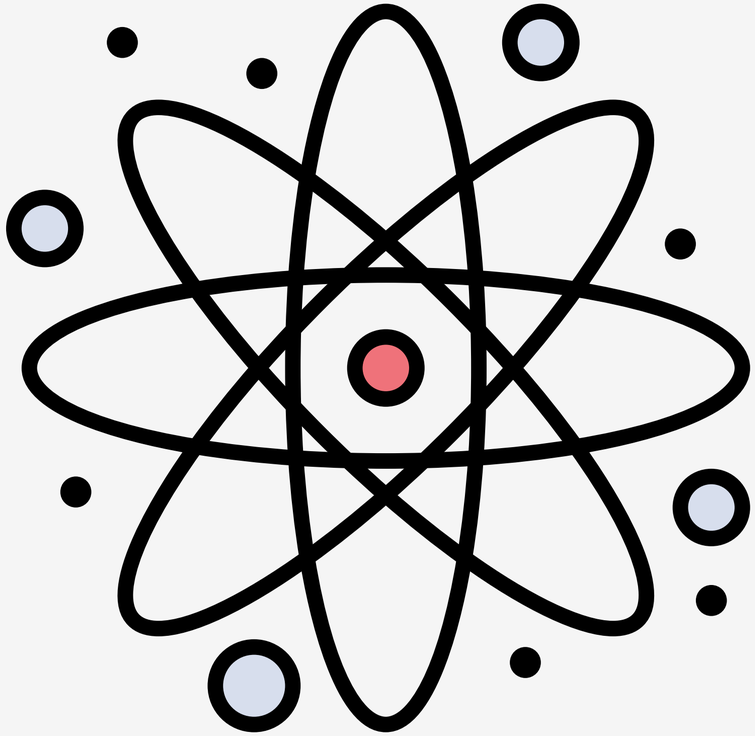


- Prepared Statements are like templates for SQL
  - Optimize database queries
  - Help prevent SQL Injection
- SQL Injection is the insertion of malicious code into a query

SQL INJECTION

```php
<?php
  require_once "db.php";
  $movie = $_GET['movie']  ';DROP TABLE movies;'

  $sql = "SELECT * FROM movies
          WHERE movie_title = '$movie'";
  $result = $db->query($sql);
```

PREPARED STATEMENTS

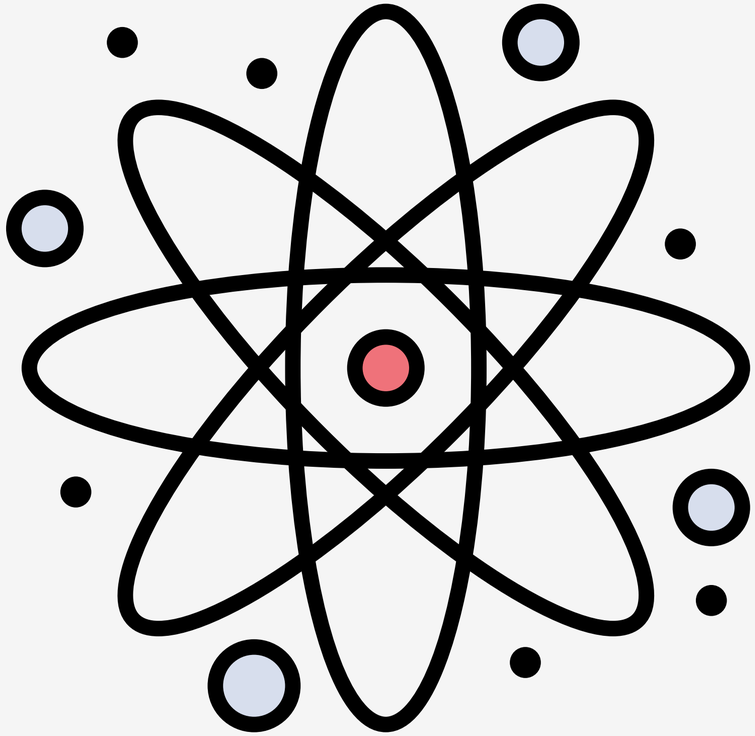


- Prepared Statements can be created using named parameters
- Named parameters serve as placeholders
- Named parameters start with a colon following by the parameter name
- The `prepare()` method tells the database to prepare the statement

PREPARED STATEMENT

```php
<?php
   require_once "db.php";
   $sql = "SELECT * FROM movies
           WHERE movie_title = :movie_title";
   $stmt = $db->prepare($sql);
```

PREPARED STATEMENTS



- Before a prepared statement can be executed, the data must be bound
- Data binding can be done using the `bindParam()`, `bindValue()` or directly in the `execute()` method
- The `bindParam()` only works with variables can is only evaluates when the statement is executed
- The `bindValue()` can be any value and can include variables

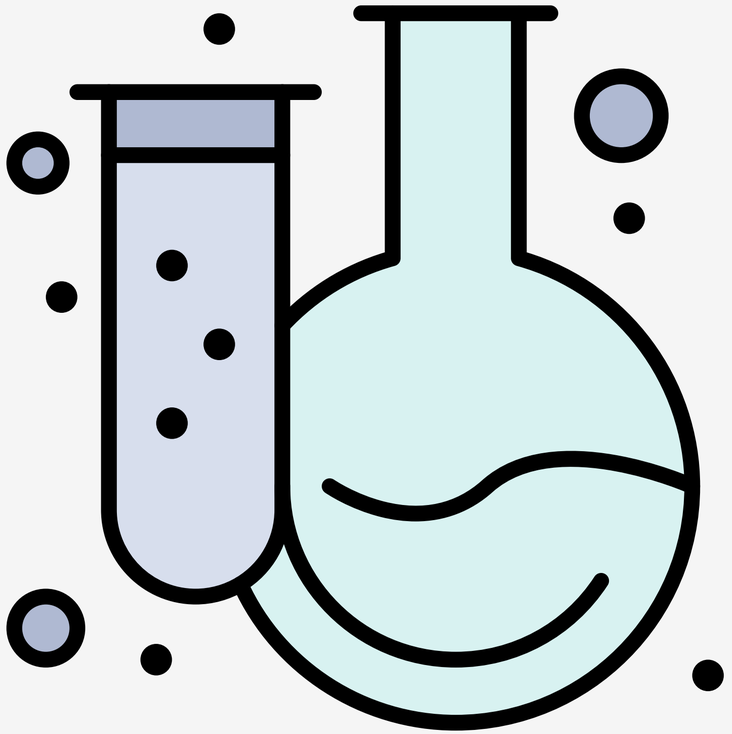
BINDING DATA

```php
<?php
   require_once "db.php";
   $sql = "SELECT * FROM movies
            WHERE movie_title = :movie_title";
   $stmt = $db->prepare($sql);
   $stmt->bindValue(':movie_title',
       $_GET['movie']);
   $stmt->execute();
```

BINDING DATA

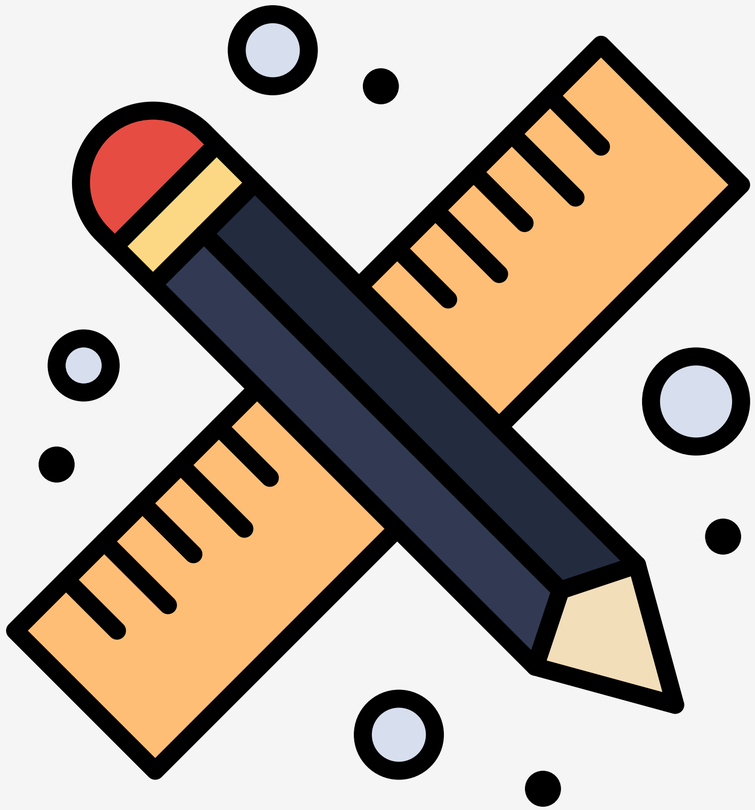
```php
<?php
  require_once "db.php";
  $sql = "SELECT * FROM movies
          WHERE movie_title = :movie_title";
  $stmt = $db->prepare($sql);
  $stmt->execute([
    ':movie_title' => $_GET['movie']
  ]);
```

HYBRID #3



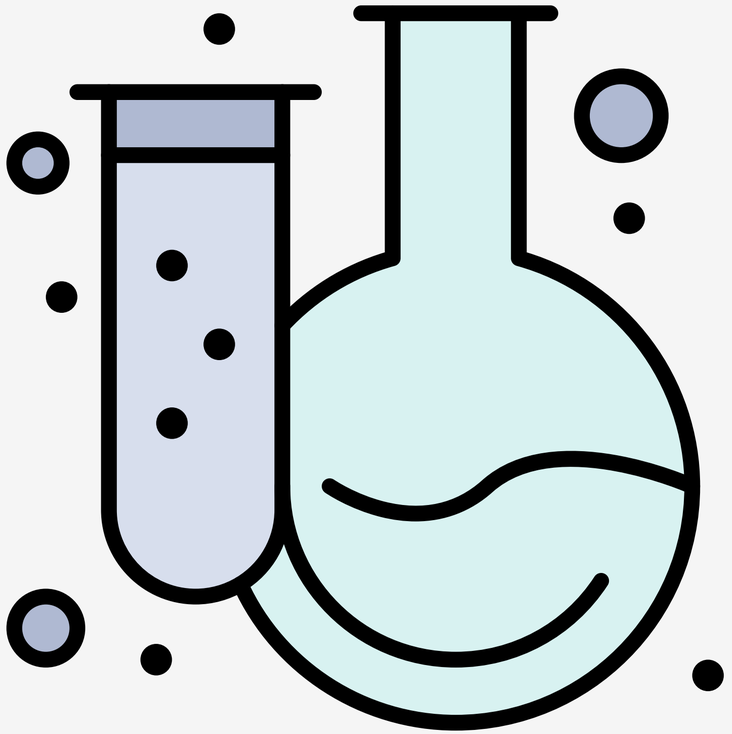
- Watch the first 4 sections of PHP: Accessing Database with PDO and MySQLi
- Write 2 to 4 sentences for each section
- *DUE:* **Wed. Jul. 8 @ 11:59 PM**

SEUSSOLOGY



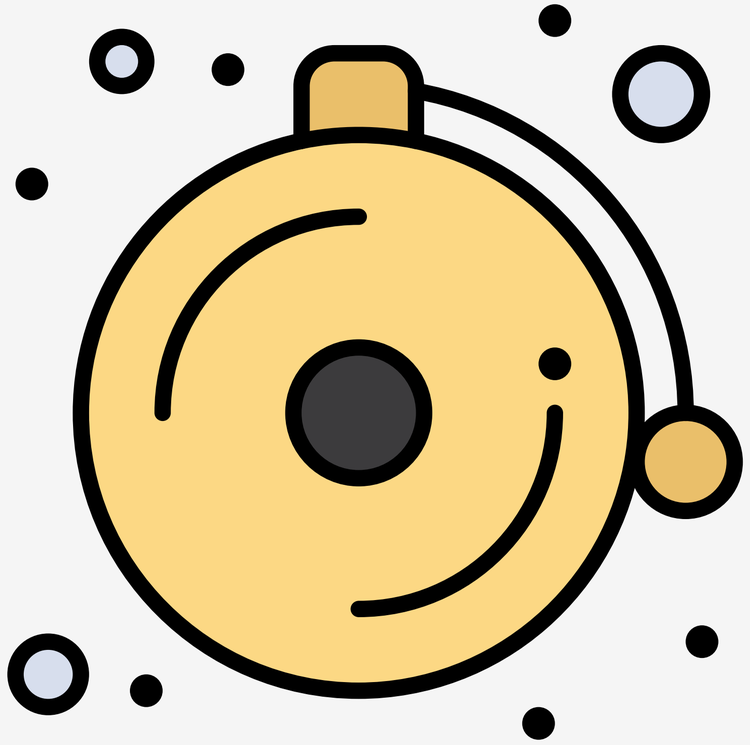
- *GITHUB CLASSROOM ASSIGNMENT*
- Import the Seussology DB - *NEW*
- Create Seussology book site
- Should be able to search books
- Add, Edit and Delete books
- *DUE:* **Wed. July 15 @ 11:59 PM**

MIDTERM REFLECTION



- Reflect and write about
  - How you completed the project
  - What problem you faced
  - What you learned
- *DUE:* **Wed. Jul. 15 @ 11:59 PM**

NEXT TIME...



- *NO CLASS NEXT WEEK*

- Laravel Routes & Views
- *DUE:* Movie Mayhem II - *TONIGHT*
- *DUE:* Seussology DB I - *TONIGHT*
- *DUE:* Hybrid #2 - *TONIGHT*
- *DUE:* Seussology DB II - Jul. 8