1. **Explain the difference between CSS Grid and Flexbox. When would you use one over the other?**

flexbox was designed for layout in one dimension - either a row or a column. Grid was designed for two-dimensional layout - rows, and columns at the same time

1. **Can you explain the DOM (Document Object Model) and how JavaScript interacts with it?**

The **Document Object Model (DOM)** is a programming interface for web documents. It represents the structure of a web page as a tree of objects, where each part of the document (like elements, attributes, and text) is represented as a node.

The DOM defines the logical or tree-like structure of the web page or document. In the tree, each branch ends in a node, and each node contains objects.

**Key Concepts of the DOM**

1. **Document Root**: The root node of the DOM tree is the document object, which represents the entire web page.
2. **Nodes**: The DOM is made up of different types of nodes:
   * **Element Nodes**: Represent HTML tags like <div>, <p>, <span>.
   * **Text Nodes**: Contain the text inside elements.
   * **Attribute Nodes**: Represent attributes like class, id, and their values.
3. **Hierarchy**: The DOM is structured like a tree:
   * Parent and child nodes define the relationships between elements.
   * Sibling nodes share the same parent.

**How JavaScript Interacts with the DOM**

JavaScript can:

* **Access Elements**: by ID, Class, using selector
* **Modify Content & Structure**:
* **Change Sty**l**es**
* **Handle Events**
* **Navigate between nodes**

1. **What is the MVC architecture, and how is it implemented in Laravel?**

MVC (Model-View-Controller) is a software design pattern used for developing user interfaces by dividing an application into three interconnected components:

1. **Model**: Represents the data and the business logic of the application. It communicates with the database, performs data validation, and handles the logic for the application.
2. **View**: Represents the presentation layer, which is responsible for displaying the data (UI). It renders the model’s data in a user-friendly format.
3. **Controller**: Acts as an intermediary between the Model and the View. It processes user requests, manipulates data through the Model, and returns the output to the View.

**MVC in Laravel**

Laravel is a PHP framework that uses the MVC architecture to separate concerns and make web applications easier to manage and scale. Here's how each part of the MVC pattern is implemented in Laravel:

**Model**:

* In Laravel, models are used to interact with the database. They represent the application's data and business logic.
* Models are typically Eloquent ORM (Object-Relational Mapping) classes in Laravel.

**View**:

* Views are stored in the resources/views directory in Laravel.
* Views are typically Blade templates, Laravel's templating engine, which allows for easier HTML rendering with embedded PHP logic.

**Controller**:

* Controllers are stored in the app/Http/Controllers directory and handle incoming requests.
* Controllers receive input from the user, process it (often interacting with the Model), and return a response (usually a View).

**Example Workflow in Laravel:**

1. A user makes a request, such as visiting http://yourapp.com/posts.
2. The request is routed to a controller (e.g., PostController@index).
3. The controller fetches data from the model (e.g., all posts from the database).
4. The controller then passes the data to the view (e.g., posts.index).
5. The view renders the HTML and displays it to the user.
6. **How do you handle database migrations in Laravel?**

In Laravel, database migrations are handled using the artisan

1. **Creating Migrations**

To create a new migration, you use the make:migration artisan command. For example:

php artisan make:migration create\_users\_table

This will generate a new migration file in the database/migrations directory with a timestamp in its filename.

1. **Writing Migrations**

Each migration file contains two main methods:

* up(): Defines the changes to be made to the database when the migration is applied (e.g., creating tables, adding columns, etc.).
* down(): Defines how to reverse the changes made in the up() method.

1. **Running Migrations**

To apply migrations, use the migrate command:

**php artisan migrate**

1. **Rolling Back Migrations**

If you need to undo a migration, you can use the migrate:rollback command:

**php artisan migrate:rollback**

1. **Explain the difference between Eloquent ORM and raw SQL queries.**
2. **What is the purpose of middleware in Laravel, and how would you use it?**

In Laravel, **middleware** acts as a filter for HTTP requests entering your application. It inspects and manipulate incoming requests before they reach the controller or outgoing responses before they are sent to the client.

Middleware is often used to handle common tasks such as:

Authentication, Authorization, Logging, Input Sanitization, Response Modification, Maintenance Mode

How to Use Middleware in Laravel :

1. Creating Middleware

php artisan make:middleware CheckAge

1. Defining Middleware Logic

Modify the handle method in the newly created middleware file:

1. Registering Middleware

 **Global Middleware**: Add it to the $middleware array in app/Http/Kernel.php. This applies the middleware to all routes.

 **Route Middleware**: Register it in the $routeMiddleware array in app/Http/Kernel.php with a key:

1. Applying Middleware to Routes

**Single Route**:

Route::get('/restricted-area', function () { return 'Welcome to the restricted area!'; })->middleware('check.age');

**Route Group**:

Route::middleware(['check.age'])->group(function () {

Route::get('/dashboard', [DashboardController::class, 'index']);

Route::get('/settings', [SettingsController::class, 'index']);

});

1. **What is REST API ?**

**RE**presentational **S**tate **T**ransfer (REST) is a web service that allows communication between a client and a server using the principles of REST architecture.

**Working:**A request is sent from client to server in the form of a web URL as HTTP GET or POST or PUT or DELETE request. After that, a response comes back from the server in the form of a resource which can be anything like HTML, XML, Image, or JSON.

In **HTTP**there are five methods that are commonly used in a REST-based Architecture i.e., POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations respectively.

1. **Explain the difference between synchronous and asynchronous code in JavaScript.**

### 1. ****Synchronous Code:****

* **Execution:** Synchronous code runs line by line, meaning each task waits for the previous one to finish before starting.
* **Blocking:** It blocks the execution of the next task until the current one is complete.

### 2. ****Asynchronous Code:****

* **Execution:** Asynchronous code does not block the execution of other code. Instead of waiting for a task to finish, it moves on to the next task, and the result of the previous task is handled later.
* **Non-blocking:** It allows the program to continue executing other code while waiting for the task to finish, which is useful for operations like network requests, timers, or file reading.

1. **Can you explain** async/await **and how it improves asynchronous code readability?**
2. **What is the purpose of break and continue statement?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **break** | **continue** |
| Effect | Exits the loop completely. | Skips the current iteration. |
| Scope | Affects the entire loop. | Affects only the current iteration. |
| Use Case | When the loop should stop. | When some iterations should be skipped. |

**break Statement**

<?php

for ($i = 0; $i < 10; $i++) {

if ($i == 5) {

break; // Exit the loop when $i equals 5

}

echo $i . " ";

}

// Output: 0 1 2 3 4

?>

**continue Statement**

<?php

for ($i = 0; $i < 10; $i++) {

if ($i % 2 == 0) {

continue; // Skip the iteration for even numbers

}

echo $i . " ";

}

// Output: 1 3 5 7 9

?>

* **Q1. What is the difference between display none and visibility hidden**
* **Q1. Given a string, write a function to reverse the string.**