Q: What are the different ways to build APIs in Drupal 10?

A: APIs can be built using core REST module, JSON:API module, GraphQL contributed module, or by creating custom REST resource plugins.

Q: What is the difference between REST, JSON:API, and GraphQL in Drupal?

A: REST provides configurable endpoints, JSON:API auto-exposes entities following JSON:API spec, while GraphQL allows flexible client-driven queries.

Q: Which core modules are required to enable REST in Drupal 10?

A: REST, Serialization, and HAL modules are required along with enabling specific entity resources.

Q: How do you enable REST for a specific entity type (e.g., node)?

A: Enable REST resource in /admin/config/services/rest and configure HTTP methods, authentication, and formats for the entity.

Q: What is serialization in Drupal? Which modules handle it?

A: Serialization converts data between PHP objects and formats like JSON/XML handled by the Serialization and HAL modules.

Q: What is the role of the Serializer and Normalizer in Drupal REST?

A: Serializer converts objects to structured data formats while Normalizer transforms entities into normalized arrays for serialization.

Q: How do you configure REST resources in Drupal 10?

A: Configure via admin UI under Web Services > REST selecting resource, methods, authentication, and formats.

Q: How do you enable authentication methods for REST (Basic Auth, Cookie, OAuth)?

A: Enable respective authentication modules and configure them in REST resource settings.

Q: What is the difference between Basic Authentication and Cookie-based Authentication in Drupal?

A: Basic Auth sends credentials with every request while Cookie authentication uses session cookies for logged-in users.

Q: How do you test REST APIs in Drupal (tools and process)?

A: Use tools like Postman or cURL to send HTTP requests and verify responses and status codes.

Q: How do you expose custom fields in REST responses?

A: Ensure the field is enabled for the entity and included in display configuration or serializer normalization.

Q: How do you restrict access to REST endpoints based on roles?

A: Configure permissions for REST resources and assign them to specific roles.

Q: Why is JSON:API preferred in Drupal 10?

A: Because it is included in core, requires zero configuration, and follows standardized API specification.

Q: How do you filter content using JSON:API?

A: Use query parameters like ?filter[field_name]=value in the endpoint URL.

Q: How do you include related entities in a JSON:API response?

A: Use the include parameter such as ?include=field_reference.

Q: How do you sort and paginate results in JSON:API?

A: Use ?sort=field_name and ?page[limit]=10&page;[offset]=0 parameters.

Q: How do you disable a specific resource in JSON:API?

A: Disable it via JSON:API resource configuration settings in admin UI.

Q: How do you create a custom REST resource in Drupal 10?

A: Create a plugin class under src/Plugin/rest/resource and register it with annotation.

Q: What annotation is used to define a REST resource plugin?

A: The @RestResource annotation defines the REST resource plugin.

Q: Which class do you extend while creating a custom REST resource?

A: Extend ResourceBase class.

Q: How do you define GET, POST, PATCH, DELETE methods in a custom REST resource?

A: Define public methods get(), post(), patch(), delete() inside the resource class.

Q: How do you handle validation inside a custom REST endpoint?

A: Validate input data inside methods and throw appropriate HttpException if invalid.

Q: How do you return custom response codes (200, 201, 403, 404)?

A: Return new ResourceResponse with status code or throw HttpException with desired code.

Q: How do you secure REST APIs in Drupal?

A: Use proper authentication, HTTPS, permissions, and limit allowed HTTP methods.

Q: How do you implement CSRF protection?

A: Enable CSRF token authentication and require X-CSRF-Token header for state-changing requests.

Q: How does caching work in Drupal REST?

A: REST responses use cache metadata including contexts, tags, and max-age for caching control.

Q: How do cache contexts and cache tags affect API responses?

A: Cache contexts vary cache per user/request context while tags allow invalidation when related content changes.

Q: How do you disable caching for a specific REST response?

A: Set max-age to 0 in ResourceResponse cache metadata.

Q: A mobile app needs to create content in Drupal. How would you implement it?

A: Enable JSON:API or REST POST endpoint with proper authentication and allow node creation permissions.

Q: How would you version your API in Drupal?

A: Implement versioning via URL prefixes like /api/v1 or create separate custom REST resources.

Q: How do you handle file/image uploads via REST?

A: Use file upload endpoint to create file entity first then reference it in content entity creation.

Q: How do you debug REST API issues in Drupal?

A: Check watchdog logs, enable REST debug logging, inspect HTTP status codes, and use Xdebug.

Q: What common REST errors have you faced and how did you fix them?

A: Errors like 403 due to missing permissions, 415 unsupported media type fixed by enabling correct format, and 401 unauthorized fixed by proper authentication setup.