Q: What is a hook in Drupal? How does it work internally?

A: A hook is a procedural callback function that allows modules to alter or extend core behavior, internally invoked via module_handler()->invokeAll() which calls all implementations following hook naming conventions.

Q: Where are hooks implemented in Drupal 10?

A: Hooks are implemented in .module files of custom or contributed modules following hook_name() naming conventions.

Q: What is the difference between procedural hooks and event subscribers in Drupal 10?

A: Procedural hooks are legacy function-based callbacks discovered automatically, while event subscribers are OOP-based Symfony EventDispatcher implementations registered as services.

Q: Why are many traditional hooks being replaced by events in modern Drupal?

A: Because events provide better OOP architecture, dependency injection, testability, and alignment with Symfony standards.

Q: What is hook discovery in Drupal?

A: Hook discovery is Drupal's runtime scanning of enabled modules to detect functions matching hook patterns and register them in the module handler cache.

Q: Explain hook_form_alter(). When would you use it?

A: hook_form_alter() modifies any form before rendering and is used when altering form structure, validation, or submission logic globally.

Q: Difference between hook_form_alter(), hook_form_FORM_ID_alter(), hook_form_BASE_FORM_ID_alter()?

A: hook_form_alter() alters all forms, hook_form_FORM_ID_alter() targets a specific form ID, and hook_form_BASE_FORM_ID_alter() alters all forms sharing the same base form.

Q: Explain hook_preprocess_HOOK() with example.

A: hook_preprocess_HOOK() alters template variables before rendering, e.g., hook_preprocess_node() to add a custom variable to node templates.

Q: What is hook_theme() used for?

A: hook_theme() registers theme hooks, templates, and render element definitions with Drupal’s theme system.

Q: Explain hook_help() and where it appears.

A: hook_help() provides contextual help text displayed on module help pages under /admin/help.

Q: What is hook_permission() used for?

A: hook_permission() defines custom permissions that can be assigned to roles in the user permissions UI.

Q: What does hook_page_attachments() do?

A: hook_page_attachments() attaches libraries, metadata, or assets to the page render array globally.

Q: What is hook_entity_presave() and when would you use it?

A: hook_entity_presave() executes before an entity is saved to modify values programmatically.

Q: Explain hook_entity_insert() and hook_entity_update().

A: hook_entity_insert() runs after a new entity is created while hook_entity_update() runs after an existing entity is updated.

Q: How would you alter a node form to add custom validation?

A: Use hook_form_FORM_ID_alter() to add a custom validation handler via $form['#validate'][] callback.

Q: How do you attach a JS/CSS library conditionally using hooks?

A: Use hook_page_attachments() or hook_form_alter() and attach library via $build['#attached']['library'][] based on condition.

Q: How would you modify a Views query using hooks?

A: Implement hook_views_query_alter() to alter the query object before execution.

Q: Explain hook_views_query_alter().

A: hook_views_query_alter() allows developers to programmatically modify SQL query conditions, joins, or filters in a View.

Q: What is hook_theme_suggestions_HOOK_alter() used for?

A: It alters template suggestion arrays to control which Twig template file is used for rendering.

Q: How do you alter route definitions in Drupal 10?

A: Use hook_route_alter() to modify existing routes or alter routing defaults.

Q: How do you alter existing services in Drupal?

A: Override services in your module’s services.yml using service decoration or altering service definitions via ServiceProvider class.

Q: What is hook_cron() and how is it triggered?

A: hook_cron() executes scheduled tasks and is triggered by Drupal cron runs via system cron or server scheduler.

Q: Which hooks are deprecated in Drupal 10 compared to Drupal 7?

A: Many menu, block, and page-related hooks like hook_menu() are removed and replaced by routing, plugins, and services.

Q: When should you use an Event Subscriber instead of a hook?

A: Use an Event Subscriber when reacting to Symfony kernel or entity events requiring dependency injection and OOP design.

Q: How do hooks interact with Symfony request lifecycle?

A: Hooks execute during Drupal bootstrap and rendering phases while Symfony events manage request-response lifecycle stages.

Q: What are cache implications when using preprocess hooks?

A: Preprocess hooks must respect cache contexts, tags, and max-age to prevent incorrect caching of dynamic data.

Q: How do you debug whether a hook is firing?

A: Use \Drupal::logger(), Xdebug breakpoints, or temporary die()/dump() statements to verify execution.

Q: In which file do you write hooks in a custom module?

A: Hooks are written inside the custom_module.module file.

Q: Can hooks be implemented inside a class in Drupal 10?

A: No, hooks must be procedural functions in Drupal 10, but Drupal 11 introduces limited class-based hook support.