ARYATECHNO

AngularJS Directives

Topics : AngularJS
Written on January 09, 2024

In AngularJS, directives are a powerful feature that extends HTML with new attributes or elements, providing a way to create reusable and modular components. Directives are one of the key building blocks of AngularJS applications, enabling developers to create custom behaviors and enhance the functionality of HTML elements.

1. **Directive Definition:** A directive in AngularJS is defined using the `directive` function, which is part of the AngularJS module. Directives can be associated with HTML elements, attributes, classes, or even comments.

   Example:

   ```
   angular.module('myApp').directive('myDirective', function() {
   return {
   restrict: 'E', // E: Element, A: Attribute, C: Class, M: Comment
   template: '<div>This is my directive</div>',
   link: function(scope, element, attrs) {
   // Directive logic goes here
   }
   };
   });
   ```

   In this example, a directive named 'myDirective' is defined, which is an element (`restrict: 'E'`). It has a simple template and a link function where the directive's logic can be implemented.

2. **Directive Usage:** Once a directive is defined, you can use it in your HTML by referencing its name as an element, attribute, class, or comment, based on the `restrict` property in the directive definition.

   Example:

   ```
   <my-directive></my-directive>
   ```

   This HTML usage corresponds to the 'myDirective' directive defined earlier.

3. **Directive Lifecycle:** Directives have a lifecycle, and AngularJS provides various hooks to

execute code at different stages. The main lifecycle hooks include:

- **compile:** This hook is used to modify the template before it is linked to the scope.
- **link:** This hook is where you can perform actions after the template has been linked to the scope.

Example:

```
link: function(scope, element, attrs) {
// Code to be executed after the directive is linked
}
```

4. **Directive Scope:** Directives can have their own scope, and you can define whether the directive should create a new scope or inherit from a parent scope. The `scope` property in the directive definition is used for this purpose.

   Example:

   ```
   scope: {
   // Define isolated scope properties here
   }
   ```

   An isolated scope ensures that the directive does not affect the parent scope.

5. **Built-in Directives:** AngularJS comes with a set of built-in directives that provide common functionality. Examples include `ng-model`, `ng-repeat`, `ng-show`, `ng-hide`, and many more. These directives enhance HTML with dynamic behavior and data binding.

   Example:

   ```
   <input type="text" ng-model="username">
   <div ng-repeat="item in items">{{ item.name }}</div>
   ```

   In these examples, `ng-model` binds an input to a variable, and `ng-repeat` iterates over an array, repeating the specified HTML for each item.