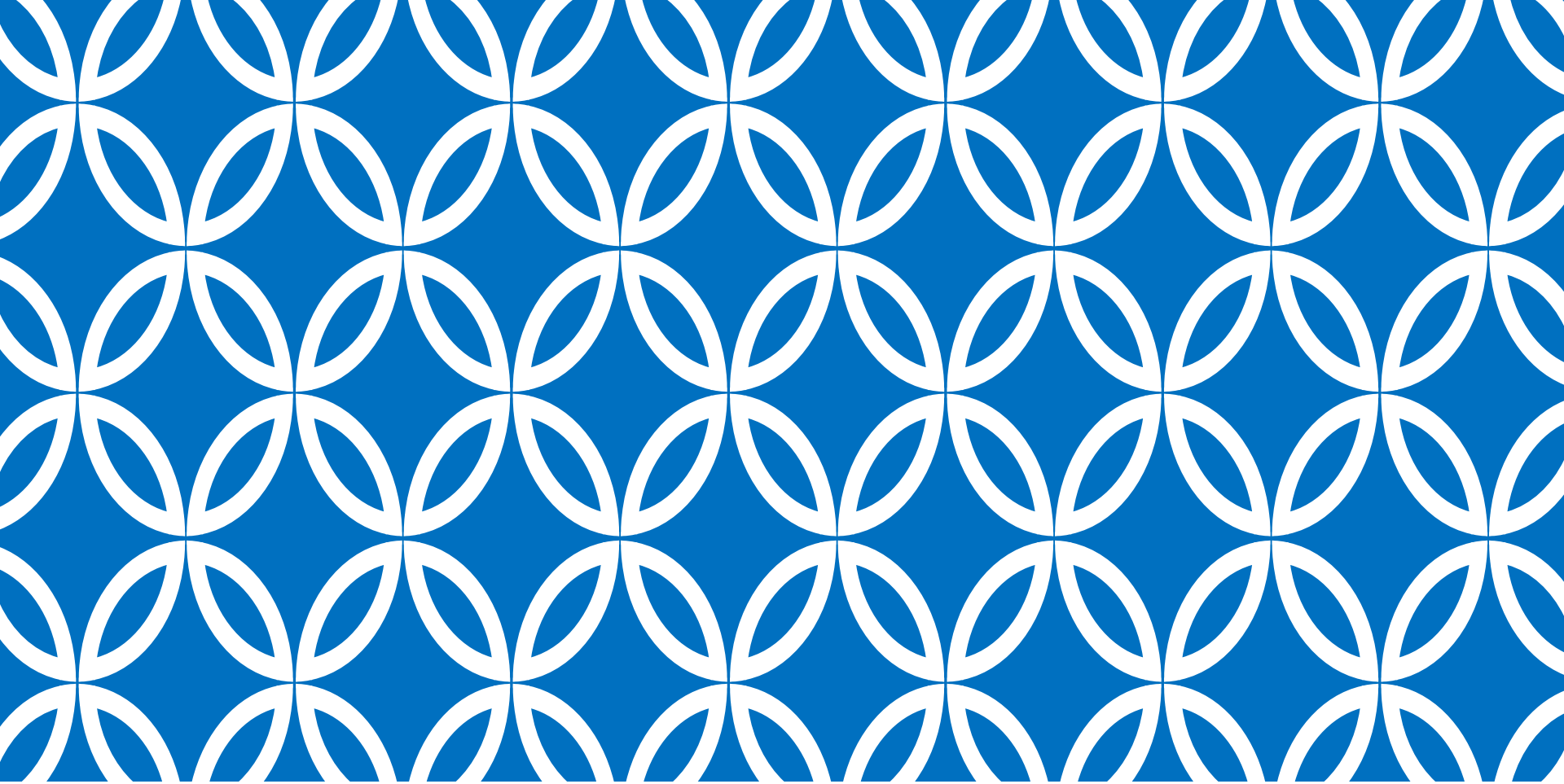
ACADGILD

FRONT END DEVELOPMENT (WITH ANGULARJS)

Website : http://www.acadgild.com
LinkedIn : https://www.linkedin.com/company/acadgild
Facebook : https://www.facebook.com/acadgild



Session 11 – AngularJS Directives

Agenda – AngularJS Directives

1. Inbuilt Directives
2. AngularJS Template
3. Sample Template
4. Understanding Scopes
5. Controllers Scopes
6. Scopes Hierarchy
7. $parent Keyword
8. this Keyword

Inbuilt Directives

- Following are the commonly used directives:
- **ng-click:** To attach a click event.
- **ng-change:** Fires a function when user changes input
- **ng-init:** to initialize data
- **ng-show:** Shows or hides the given HTML element based on the expression provided to the ngHide attribute
- **ng-href:** To point a URL for a link

Inbuilt Directives

- **ng-include:** This directive fetches, compiles and includes an external HTML snippet.
- **ng-model:** to bind a form input .
- **ng-repeat:** This directive is used to repeat the html for a particular collection. Useful to show an array items in a list etc.
- **ng-selected:** If the expression value is true, then special attribute "selected" will be set on the specified element.
- **ng-show:** It shows or hides the given HTML element based on the expression provided to the ng-show attribute.

AngularJS Template

- AngularJS templates are just plain old HTML that contains Angular-specific elements and attributes.

- AngularJS used these templates to show information from the model and controller.

- Inside your AngularJS templates you can define following angular elements and attributes:
  - Directive
  - Angular Markup ({{ }})
  - Filters
  - Form controls

Sample Template

Template

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Title
    </title>
    <meta charset="UTF-8" />
    <style media="screen"></style>
    <script src="angular.min.js"></script>
  </head>
  <body>
    <!-- initialize the app -->
    <div ng-app>
      <!-- store the value of input field into a variable name -->
      <p>Name: <input type="text" ng-model="name"></p>
      <!-- display the variable name inside (innerHTML) of p -->
      <p ng-bind="name"></p>
    </div>
  </body>
</html>
```

Download this file from:
https://angularjs.org/

Directive

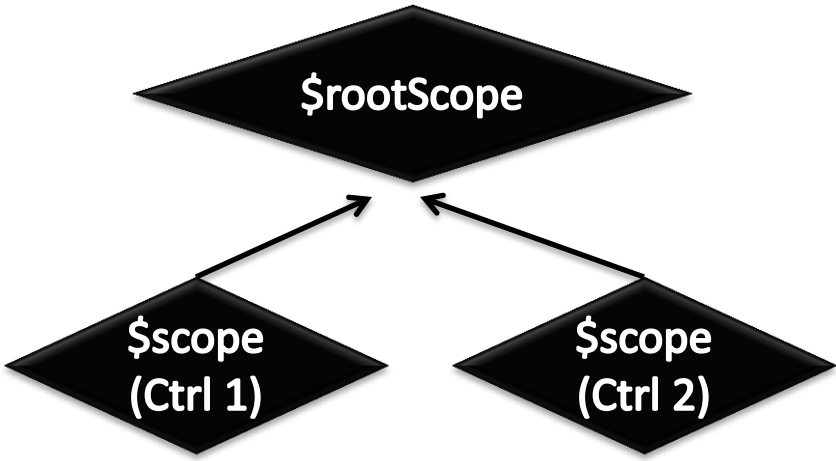
Directive

Understanding Scopes

- Scope is an object that refers to the application model.
- It acts as a context for evaluating expressions.
- Typically, it acts as a glue between controller and view.
- Scopes are hierarchical in nature and follow the DOM structure of your angular app.
- AngularJS has two scope objects: **$rootScope** and **$scope**.

Controllers Scopes

- A $scope is a JavaScript object which is used for communication between controller and view.
- Basically, $scope binds a view (DOM element) to the view model and functions defined in a controller.
- The **$rootScope** is the top-most scope.
- An app can have only **one $rootScope** which will be shared among all the components of an app. Hence it acts like a global variable.
- All other $scopes are children of the $rootScope.



Scopes Hierarchy

- **What happens if we keep one controller within another controller?**
  - The controller that has other controller inside becomes the parent controller. Now the child controller will inherit all the properties of the parent controller.
  - Any properties of the parent controller can be over written.

$parent Keyword

- Angular scopes include a variable called **$parent** that refers to the parent scope of a controller.

- If a controller is at the root of the application, then parent would be the root scopei.e. $rootScope

- There are instances when we have overwritten the property of the parent controller within child controller and we still want to access or modify it. In this case we can use the $parent keyword.

this Keyword

- We can use $scope injectible to give access to the data within the html. Though this works fine, we can also use **this** operator of the controller instance to achieve the same.

- The benefit of using **this** operator is that we don't need to use the $scope to attach data to the scope.

Lets Discuss Assignments