Different properties of directive :

a1)restrict a2)scope a3)template a4)templateUrl a5)controller a6)link

Use it with attribute | ‘E’, ‘A’, class | restrict : ‘C’, comment | ‘M’ | replace : ‘true’

restrict  |template  | replace

Ref : https://www.sitepoint.com/practical-guide-angularjs-directives/

* Note :replace – This specifies if the generated template will replace the HTML element on which the directive is attached. In our case we have used the directive as <hello-world></hello-world>, and replace is set to true. So, after the directive is compiled, the produced output template replaces <hello-world></hello-world>. The final output is <h3>Hello World!!</h3>. If you set replace to false, the default, the output template will be inserted into the element on which the directive is invoked.

9a)Create a new custom directive with company details .

employee\_id, employeefirstname ,employeelastname, salary

9a1)Create a custom directive pass length f3 side of the triangle.Display the area of the triangle.

9a2)

9a3)Create a simple directive with link events as mouseover,mouseout.Change the color of paragraph tag.

9a4)Change image src on mouseover and out.

Inside the mouseover do scope.$apply to change the scope value.

## Note : $apply enables to integrate changes with the digest cycle. <http://stackoverflow.com/questions/15112584/how-do-i-use-scope-watch-and-scope-apply-in-angularjs>

Note **:** When that button is clicked the $scope.updateTime() function is called, and after that AngularJS calls $scope.$digest() so that data bindings are updated.

**Note** : You can think of the $apply **function as of an integration mechanism**. You see, each time you change some **watched variable attached to the** $scope object directly, AngularJS will know that the change has happened. This is because AngularJS already knew to monitor those changes. So if it happens in code managed by the framework, the digest cycle will carry on.

However, sometimes you want to **change some value outside of the AngularJS world** and see the changes propagate normally. Consider this - you have a $scope.myVar value which will be modified within a jQuery's $.ajax() handler. This will happen at some point in future. AngularJS can't wait for this to happen, since it hasn't been instructed to wait on jQuery.

To tackle this, $apply has been introduced. It lets you to start the digestion cycle explicitly. However, you should only use this to migrate some data to AngularJS (integration with other frameworks), but never use this method combined with regular AngularJS code, as AngularJS will throw an error then.

9a5)Also use scope.$watch for checking any change in the scope variable last\_name. If yes then display some message with console.log

elem.bind('click', function() {

elem.css('background-color', 'white');

scope.$apply(function() {

scope.color = "white";

});

9a2)Add the scope.$apply for background-color,color on mouseover and out.

Also give pointer (hand) on mouseover.

element.css(‘cursor’,’pointer’)

Note :The link function is mainly used for attaching event listeners to DOM elements, watching model properties for changes, and updating the DOM.

9a3)Write a directive such that initialize the $scope.carsPrint every cars in li tag with strong tag through directive. Take restrict as ‘A’,’C’

Css : font-size , color

<li ng-repeat="x in movies">

<my-directive name='{{x}}'></my-directive>

</li>

</ul>

First : use the ‘@’ scope tag

Ref :<https://egghead.io/lessons/angularjs-isolate-scope-two-way-binding>

9a5)Write a script by making use of $watch . Create a form for the employee containing emp\_first\_name, emp\_last\_name, emp\_sal . Add button so that it can be pushed to the array.When the size os array increases greater then 5 the make the table background as red.

**Isolate scope start**

*Ref :<https://thinkster.io/egghead/isolate-scope-am>*

*Ref :https://www.youtube.com/watch?v=STKRRuSkL3c*

All are isolate scope :

* @ - pass by value (attribute binding)
* & - pass by method (expression binding)
* = - pass by reference (two-way data binding)

Q1) First make a directive without the isolate scope. For this take $scope.a =10. Display the a in the template of directive as **template : ‘Value is {{a}}’.** There is now scope {} applied upto now.Once scope {] applied the value os a gets disappear.

Ref : pankaj\_git\imp\_items\angular\_1\_directive\ scope\_check.html

Q2)Now apply scope{}.Then check the output.

Ref : pankaj\_git/imp\_items/angular\_1\_directive/scope\_check2.html

Isolate Scope @ : Pass by value.

Q3a)Now use **Isolate @.** Check for result. This is also called pass by value.

Ref : pankaj\_git/imp\_items/angular\_1\_directive/scope\_check3.html

Q3b)Refer the variable with different name

Ref : pankaj\_git/imp\_items/angular\_1\_directive/scope\_check5\_2.html

Isolate Scope = : Pass by reference

Q4a)Use **Isolate = .** This is pass by reference. It also have two way binding.

Ref : pankaj\_git/imp\_items/angular\_1\_directive/scope\_check4.html

Q4b)Again make a textfield to show 2way data binding.

Ref : pankaj\_git/imp\_items/angular\_1\_directive/scope\_check5.html

Isolate Scope & : Pass by method

Q1)Create a product object in the controller. Display the object with json object in **p** tag in the application body. Now create the directive with pass by reference for the product.Change in product title will change the controller json. Also pass the product object to showProduct() function.

Ref : pankaj\_git/imp\_items/angular\_1\_directive/button\_pass\_method.html

Isolate scope end

9a3)angular directive compile phase : <https://www.bennadel.com/blog/2794-when-do-you-need-to-compile-a-directive-in-angularjs.htm>

Note :link and compile do not work together.

In the directive definition object, if you **only** define link, that's like shorthand for having an empty compile function with an empty preLink function with your code in the postLink function. As soon as you define compile, link is ignored by angular, because compile should return the linking functions.

If you only return one function from compile, then it'll be executed **post** link.

Or, put differently, link is just a shortcut to the postLink function that gets called after the scope has been linked by compile.

Updating last line..

There are 3 types of angular directives.

1)Components : They are custom element also called the widgets.It contains it’s own html/template. It is a self oriented html.

2)Decorators : To enhance the functionality of existing directive. ng-click , ng-hide , ng-show. They do not have their own template

3)Templating : Structural based directive. They changes the structure of the html. ng-repeat

Q1)Display the employee recird on the directive.Search employee and displaying it on directive also display complete employes with the help of the directive:

https://www.youtube.com/watch?v=ubigWmtTzQM

**Compile and link in depth : https://www.youtube.com/watch?v=bjFqSyddCeA**

Compile : First set of execution/processing a directive for rendering.

Example :

<div ng-controller=”msg”>

<div message text=”first”>..</div>

</div>

So angularjs just first tries to find our directive **message** and then tries to compile it. Compilation takes place regardless of the buildin custom directive or predefined directive.

**Compile function** : If we want some of the code execution at the part of compilation phase. It happens only once. It loads and traverse the template. Same template never compiled more then once. Compile function returns a link function or an object which performs the pre , post . No scope is present as a part of compilation process. We can not access the scope during the compilation process.

No instance or the clone of the template created yet. This is the part of linking process. No data is maintained or updated during the compilation process. This is because we do not have any scope here. Suppose we applied the border color then same view is available for all the clones for the directive template that we have managed during the compilation process.So that same DOM can be cloned.

Note : Again saying we can not play with the data/events. No events or event handles during this phase.

compile(tElement,tAttributes){}

Note : <div message text=”first”>..</div> complete refers to the tElement.

text=”first” this refers to the tAttributes

**Link** : It is the next phase after the compilation process. It has many phases.

As a part of link first whole **<div message text=”first”>..</div>** item gets cloned into the memory. This instance or clone is created just after the compilation process. Linking then proceed with this instance that got created into the memory.

Lining has the 3 major steps.

a)**Controller** : If it is available it is executed first.Every directive has it’s own controller associated with it.Once the template has been cloned the controller gets kicking in. In this phase the scope gets created. Here we can modify the data. Can any type os variables , properties , method to that scope. We can also access the DOM of the template. But it is not good to change the css of template.

b)**Pre Link** : It is the just next step of the controller. It also gets executed on the same template instance. The DOM template is available during the pre-link phase. The scope that get available at the controller phase still available in the pre link phase. User can access the scope related data here. The scope generated during the controller and that is available in pre link is not bound completely with the DOM template. Child directives are not processed at this moment. Can not access any of the child directive. Also not save to change the DOM template.

c)**Post Link :** It works on the same instance . It is the last phase before the render. Once the post link gets completed , it is going to render finally. Here DOM gets prepared completely . Means completely bound to the scope. Here we can manipulate the DOM. All the child directives are ready. We can access the scope but not required in this phase . We can attach any type of the listener for dom elements here. Post link is best place for applying the click event for the button. No save to save the data for the child elements. Data for the child element is best provided during the Pre link.

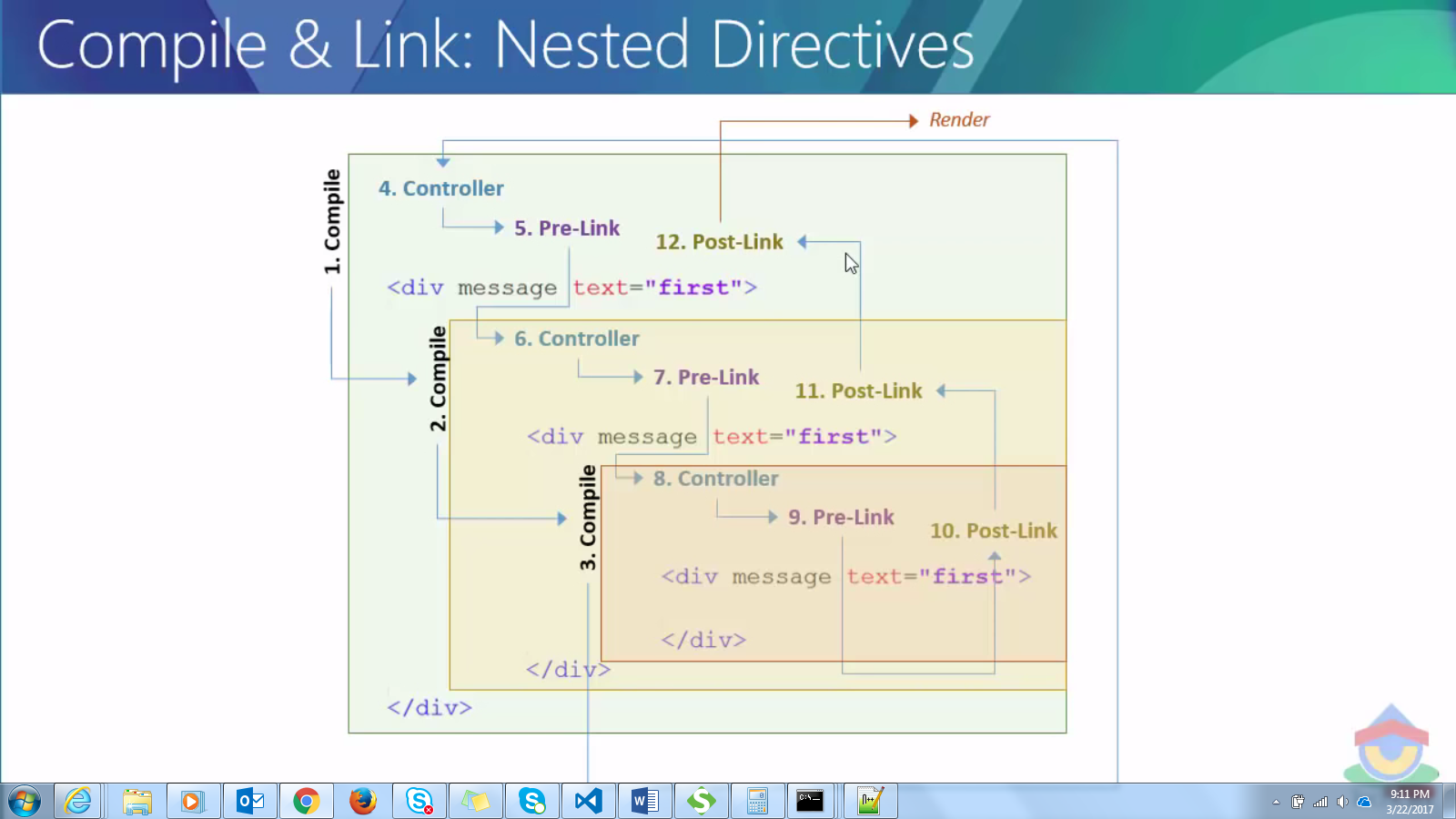
All the 3 steps are performed against the instance or the clone created into the memory.

Note : Once the linking gets completed the angularjs render that clone created on html view.

The execution of compile and the link:

**Nested directive :**

How the compile and link will work for the nested directive.



First the compilation of Our directive , then inner directives takes place. All the compilation for all directives needed to complete first before going to the other phase.

After this the control goes to the linking process. Controller invoked first of them.

**Q1)Write a programme to display the flow from the compile function to the post function of link.**

**Q2)Write a programme to display the flow from the compile function to the post function of link.Do it for the nested directive.**

Output hint :

Inside compile function First directive

Inside compile function Second directive

Inside compile function Third directive

Inside the controller function First directive

Inside pre function First directive

Inside the controller function Second directive

Inside pre function Second directive

Inside the controller function Third directive

Inside pre function Third directive

Inside post function Third directive

Inside post function Second directive

Inside post function First directive

**Ref : pankaj\_git\imp\_items\angular\_1\_directive\** **life\_cycle\_directive.html**

Carry on for part 2 : [**https://www.youtube.com/watch?v=1M3LZ1cu7rw**](https://www.youtube.com/watch?v=1M3LZ1cu7rw)

**$watch concept : https://www.youtube.com/watch?v=VS6vDlsqW7o**