**PreLink, PostLink and Controller Methods of Angular Directives**

## Link function of an Angular Directive

As the name implicates, the link function has the duty of linking the model to the templates. Link function is the place where AngularJs does the data binding to the compiled templates. Let’s take a look at the signature of a link function.

|  |  |
| --- | --- |
| 1 | link: function LinkFn(scope, elem, attr, ctrl){} |

There are 4 parameters available to the link function.

1. **scope** : The scope of the directive
2. **elem** : Dom element where the directive is applied
3. **attr** : Collection of attributes of the Dom Element
4. **ctrl** : Array of controllers required by the directive

Now let’s create a simple directive to see how the data binding works. See the JSFiddle below:

var app = angular.module('app', []);

app.directive('dad', function () {

return {

restrict: 'EA',

template: '<div>{{greeting}}{{name}}</div>',

link: function(scope,elem,attr){

scope.name = 'Paul';

scope.greeting = 'Hey, I am ';

}

};

});

<div ng-app="app">

<dad></dad>

</div>

[**http://jsfiddle.net/shidhincr/Bpxn2/light/**](http://jsfiddle.net/shidhincr/Bpxn2/light/)

The name and greeting properties attached to the scope are linked to the template once the link function is executed. And, the browser will show **“Hey, I am Paul”** in the view.

The above is the usual way to create a link function inside a directive. However, AngularJs allows to set the link property to an object also. Advantage of having an object is, we can split the link function into two separate methods called, pre-link and post-link. In the following sections, we’ll see how to use these link functions.

## PostLink

In the previous sections, we saw how to create a link function. For AngularJs, the link function is we created is a post-link function. So in general we can write the post-link function in two ways:

**1)** Simply set the link method.

var app = angular.module('app', []);

app.directive('dad', function () {

return {

restrict: 'EA',

template: '<div></div>',

link: function(scope,elem,attr){

scope.name = 'Paul';

scope.greeting = 'Hey, I am ';

}

};

});

**2)** link property points to a object literal, which has a post method.

var app = angular.module('app', []);

app.directive('dad', function () {

return {

restrict: 'EA',

template: '<div></div>',

link: {

post: function(scope,elem,attr){

scope.name = 'Paul';

scope.greeting = 'Hey, I am ';

}

}

};

});

## PreLink

The signature is of the pre-link function is same as that of a post-link. The only difference between the pre-link and a post-link is the order they gets executed. The following code will explain more clearly.

Let’s create a new directive called <son> and place inside the template of <dad> directive.

var app = angular.module('app', []);

app.directive('dad', function () {

return {

restrict: 'EA',

template: '<div class="dad">{{greeting}}{{name}}'+

'<son></son>'+

'</div>',

link: function(scope,elem,attr){

scope.name = 'Paul';

scope.greeting = 'Hey, I am ';

}

};

});

app.directive('son', function () {

return {

restrict: 'EA',

template: '<div class="son">{{sonSays}}</div>',

link: function(scope,elem,attr){

scope.sonSays = 'Hey, I am son, and my dad is '+ scope.name;

}

};

});

We created a **son** directive and placed inside the **dad** directive’s template. Since there is no scope specified for the **son** directive, we assume that all parent directive scope should be available to it. Let’s look at the output tab of the jsFiddle, we can see that the **son** directive prints like this:

Hey, I am son, and my dad is undefined

Notice that the dad’s name is undefined ?

Now let’s analyse what happened. Here, both the **dad** and **son** directives have link functions, and both these link functions are post-links. When a directive contains multiple child directives, all of the child directive’s link functions executed first then the parent directive link function. So, in this case, when **son** directive’s link function executes, the **dad** directive is still not linked the data to the template. That’s why the **son** directive outputs the dad’s name as **undefined**.

<http://jsfiddle.net/shidhincr/Bpxn2/1/light/>

How to solve this issue ?

This is where the pre-link comes handy. A pre-link function of a directive will get executed before all of its child directives’ link functions. Let’s modify our jsFiddle:

var app = angular.module('app', []);

app.directive('dad', function () {

return {

restrict: 'EA',

template: '<div class="dad">{{greeting}}{{name}}'+

'<son></son>'+

'</div>',

link: {

pre: function(scope,elem,attr){

scope.name = 'Paul';

scope.greeting = 'Hey, I am ';

}

}

};

});

app.directive('son', function () {

return {

restrict: 'EA',

template: '<div class="son">{{sonSays}}</div>',

link: function(scope,elem,attr){

scope.sonSays = 'Hey, I am son, and my dad is '+ scope.name;

}

};

});

<http://jsfiddle.net/shidhincr/Bpxn2/2/light/>

#### **As per the Docs**

#### **Pre-linking function**

Executed before the child elements are linked. Not safe to do DOM transformation since the compiler linking function will fail to locate the correct elements for linking.

## Controller

Changing post-link to pre-link will solve the above problem. However, it’s not a best practice to create pre-link functions whenever we introduce a child directive. Assume, if instead of a child directive, what if we want to share some data to another directive applied to the same DOM element ?

Directive’s controller is designed for that. A controller is a place where directive can define it’s public API. Let’s solve the above problem in controller way.

var app = angular.module('app', []);

app.directive('dad', function () {

return {

restrict: 'EA',

template: '<div class="dad">{{greeting}}{{name}}'+

'<son></son>'+

'</div>',

controller: function(){

this.name = 'Paul'

},

link: function(scope,elem,attr,ctrl){

scope.name = ctrl.name;

scope.greeting = 'Hey, I am ';

}

};

});

app.directive('son', function () {

return {

restrict: 'EA',

require: '^dad',

template: '<div class="son">{{sonSays}}</div>',

link: function(scope,elem,attr,ctrl){

scope.sonSays = 'Hey, I am son, and my dad is '+ ctrl.name;

}

};

});

<div ng-app="app">

<dad></dad>

</div>

<http://jsfiddle.net/shidhincr/Bpxn2/3/light/>