alight Documentation

Release 0.10.0

Oleg Nechaev

November 16, 2015

Contents

Contents:

CHAPTER 1

API

1.1 alight.debug

Turn on a logging debug information

- alight.debug.scan - logging scan operations
- alight.debug.directive - logging binding directives
- alight.debug.parser - logging parsing expressions
- alight.debug.watch - logging function $watch
- alight.debug.watchText - logging function $watchText

1.2 alight.autostart = false

Don't call alight.bootstrap on start

1.3 alight.bootstrap(element)

Manual start applications, element - DOM element or array of elements. If element is undefined then Angular Light will start all al-app on the document. If alight.autostart = true, the method will be called after the DOM is ready

1.4 alight.applyBindings(scope, element, config)

Bind the scope to the DOM element

- scope - Scope object
- element - DOM element
- config.skip_top = false - Skip binding the top DOM element
- config.skip_attr = ['al-repeat', 'al-app'] - Skip attributes for a binding, for the top element

```
var scope = alight.Scope();
var element = document.body;
alight.applyBindings(scope, element);
```

1.5 alight.Scope()

Create a (root) Scope

1.6 alight.controllers

Dictionary of controllers

1.7 alight.filters

Dictionary of filters

1.8 alight.directives

Dictionary of directives

1.9 alight.nextTick(callback)

Execute the function on next tick

1.10 alight.getController(name, scope)

Take a controller by name. A controller can be located in alight.controllers, in global scope

CHAPTER 2

Scope

2.1 Scope.$watch(name, callback, option)

Set the tracking variable. Also you can track system events, it returns object with method stop()

Name:

- **<expression>** - Expression/model
- **<a function>** - Track the result of the function, the function are called every iteration of $scan.
- **“$destroy”** - Track a destroying scope
- **“$any”** - Track a modifying any object
- **“$finishScan”** - a callback is executed as soon as $scan finish work
- **“$finishBinding”** - the callback is called when a binding process finishes, sample
- **“$finishScanOnce”**

Option:

- **option** = true or **option.isArray** = true - watch an array
- **option.init** = true - Execute *callback*
- **option.readOnly** = true - You can use it if the *callback* doesn’t modify the scope. (an optimization option).
- **option.deep** = true - a deep comparison for the object.
- **option.isArray**
- **option.OneTime**
- **option.onStop**

Optimization tip: If *callback* returns ‘$scanNoChanges’ then $scan will not run extra-loop (like readonly watch)

2.2 Scope.$compile(expression, option)

Compile an expression

option:

- **option.input** - list of input arguments
- **option.no_return** - a function will not return any result (compile a statment)

- **option.string** - result of method will convert to string

Listing 2.1: Example of $compile

```
var scope = alight.Scope()
var fn = scope.$compile('"hello " + title')
scope.title = 'linux'
fn() // return "hello linux"
scope.title = 'macos'
fn() // return "hello macos"
```

Listing 2.2: Example with input

```
var fn = scope.$compile('title + v', { input:['v'] })
fn(' X') // return "macos X"
```

Listing 2.3: Example with no_return

```
var fn = scope.$compile('title = v', { input:['v'], no_return:true })
fn('linux') // scope.title = "linux"
```

2.3 Scope.$eval(expression)

Execute an expression

2.4 Scope.$watchText(tpl, callback)

Track the template

2.5 Scope.$compileText(tpl)

Compile the template. Method is depricated (since v0.9)

2.6 Scope.$evalText(tpl)

Evalute the template. Method is depricated (since v0.9)

2.7 Scope.$new(isolate)

Create a child Scope, if isolate == true, then child scope will not be inherited from parent scope, if isolate == 'root' then it will be separate root scope.

2.8 Scope.$destroy()

Destroy the Scope.

Listing 2.4: Example with $complieText

```
var scope = alight.Scope()
var fn = scope.$compileText('Hello {{title}}!')
scope.title = 'linux'
fn() // return "Hello linux!"
```

2.9 Scope.$scan(callback or option)

Start the search for changes

- **callback** - Method will be called when $scan finishes a work, even if $scan has already started from other a place.
- **option.callback** - see above.
- **option.top** - Choose the root scope for current scanning (depricated).
- **option.late** = *(true/false)* - If there is a few $scan commands, Angular Light will call only last one.

Listing 2.5: Example with $scan

```
var scope = alight.Scope()
scope.$watch('title', function(value) {
    console.log('title =', value)
}) // make observing
scope.title = 'new'
scope.$scan()
// print title = new
scope.title = 'linux'
scope.$scan()
// print title = linux
scope.$scan()
// do nothing
```

2.10 Scope.$scanAsync(callback)

It the same as *Scope.$scan({late: true, callback: callback})*

2.11 Scope.$getValue(name)

Take the value of the variable, also you can use Scope.$eval

2.12 Scope.$setValue(name, value)

Set the value of the variable

Listing 2.6: Example with $setValue

```
scope.var = 1;
scope.path.var = 2;
scope.path[scope.key] = 3;

// equal
scope.$setValue('var', 1);
scope.$setValue('path.var', 2);
scope.$setValue('path[key]', 3);
```

CHAPTER 3

Create Directives

Directives should be placed in **alight.directives.prefix**, you can choose any prefix, for example al-text (*alight.directives.al.text*), bo-text (*alight.directives.bo.text*), also you can place your directives in *scope.$ns.directives* they will be private.

A hyphen should be changed to underscores, example: `<input al-my-some-directive />` it will be in alight.directives.al. **mySomeDirective**

The prefixes are needed in order to be able to catch the lack of used directives. For example, if you use the directive “al-texta”, but it does not exists (a mistake in the name or js-file isn’t loaded), then aLight will throw an exception.

An example of directive **al-text**, the directive is called when the binding process comes to an DOM-element

Listing 3.1: Example of directive al-text

```
alight.directives.al.text = function(element, name, scope, env) {
    // a function to set a text to the DOM-element
    var setter = function(text) {
        $(element).text(value)
    }

    // Track to the variable
    scope.$watch(name, setter, {
        init: true  // Run the setter immediately
    });
};
```

Input arguments:

- **element** - element of DOM
- **name** - value of attribute
- **scope** - current Scope
- **env** - access to different options
- env. **attrName** - name of attribute (directive)
- env. **attributes** - list of attributes
- env. **takeAttr(name, skip=true)** - take a value of the attribute, if skip=true then the attribute will skip a binding process, sample
- env. **skippedAttr()** - list of not active attributes.

3.1 Attributes of directive:

- **priority** - you can set priority for a directive
- **template** - custom template
- **templateUrl** - link to template
- **scope** (true/'isolate'/'root') - to make a child scope, an isolated scope (with no inheritance), a separate root scope
- **restrict** = 'A', can be 'AEM', 'A' matches attribute name, 'E' matches element name, 'M' matches class name
- **init** - the method is called when the directive is made, before template, scope
- **link** - the method is called after template, scope
- *anyAttr* - you can make a custom attribute, look. directive preprocessor

Listing 3.2: Directives with attributes

```
alight.directives.al.stop = {
    priority: -10,
    template: '<b></b>',
    scope: true,
    init: function(element, name, scope, env) {
        return {
            owner: true
        };
    },
    link: function(element, name, scope, env) {
    }
};
```

If a directive returns the flag owner `return { owner:true }`, then the process of binding will miss child DOM-elements, it is necessary for the directives which are themselves controlled subsidiary of DOM, such as al-repeat, al-controller, al-include, al-stop, etc.

CHAPTER 4

Inheritance of directives

If you want make inheritance of a directive, you need call a parent directive after that you can replace methods of the directive. For example, **al-value** has a few methods:

- onDom - binding to DOM
- updateModel - updateing the model
- watchModel - $watch model
- updateDom - updateting DOM
- initDom - set first value to DOM, updateDom(init_value)
- start - It's called when the directive is ready

Make a directive al-value with deferred updating of model:

Listing 4.1: Inherit al-value

```
alight.directives.al.valueDelay = function() {
    // create a parent directive
    var dir = alight.directives.al.value.apply(null, arguments);

    // save the old method for update the model
    var oldUpdate = dir.updateModel;
    var timer = null;

    // change the method
    dir.updateModel = function() {
        if(timer) clearTimeout(timer);
        timer = setTimeout(function() {
            timer = null;

            // call the default method for update the model
            oldUpdate();
        }, 500);
    }
    return dir;
}
```

4.1 Examples of inheritance

- al-value -> al-value-delay

- al-show -> al-show-slow
- al-repeat, add “$even”, “$odd” into the directive
- al-repeat, change input expression my-repeat=”list.foreach item”
- al-repeat, change rendering

CHAPTER 5

Directives

5.1 Events

- al-click, todo sample
- al-dblclick
- al-submit, todo sample
- al-blur, Sample
- al-change, Sample
- al-focus, Sample
- al-keydown, Sample
- al-keypress, Sample
- al-keyup, Sample
- al-mousedown, Sample
- al-mouseenter, Sample
- al-mouseleave, Sample
- al-mousemove, Sample
- al-mouseover, Sample
- al-mouseup, Sample

5.2 Controls

- al-checked, todo sample
- al-radio sample1 sample2
- al-value, todo sample
- al-disable
- al-enable
- al-focused, two-way bind for focus events. Sample
- al-readonly

5.3 Special directives

- al-app, init application with current element, examples
- al-cloak, hide current element until activate the application, examples
- al-controller, apply a custom controller and make child scope for current element.
- al-class/al-css, todo sample, animated sample
- **al-style** examples
- al-hide, sample with animation
- al-html
- **al-if**, sample with animation
- al-ifnot, sample with animation
- **al-include**, loads a html block from the server, sample with animation
- al-init
- al-repeat
- **al-show**, sample with animation
- al-src
- al-stop, stops a bind process for the element and his children.
- al-text, example

5.4 Bind-once

- bo-if
- bo-ifnot
- bo-repeat
- bo-src
- bo-switch
- bo-switchDefault
- bo-switchWhen

CHAPTER 6

Base filters

6.1 data

- To convert the date to string
- Input argument: date

Listing 6.1: example

```
<div>{{when | date:yyyy-mm-dd }}</div>
```

6.2 filter

- To filter the list
- Input argument: variable

Example

6.3 slice

- To slice the list
- input arguments: numbers

Listing 6.2: example

```
<div al-repeat="it in list | slice:a"></div>
<div al-repeat="it in list | slice:a,b"></div>
```

6.4 generator

- The filter makes an array (for al-repeat)
- input arguments: numbers

Example

Listing 6.3: example

```
<div al-repeat="it in 10 | generator"></div>
<div al-repeat="it in size | generator"></div>
```

6.5 toArray

- converts an object to array (for al-repeat)
- input arguments: key, value

Listing 6.4: example

```
<div al-repeat="item in object | toArray:key,value track by key">
```

Example

6.6 orderBy

- sorts an array by key (for al-repeat)
- input arguments: key, reverse

Listing 6.5: example

```
<div al-repeat="item in array | orderBy:key,reverse">
```

Example

6.7 throttle

- makes delay for output, pattern *debounce*
- input arguments: delay

Example

Listing 6.6: example

```
<input al-value="link" type="text" />
<p>{{link | throttle:300 | loadFromServer}}</p>
```

CHAPTER 7

Create filters

7.1 Overview

A filter should be placed in **alight.filters** (you can change path with *alight.getFilter*), or in scope.$ns for private filters.

Listing 7.1: Example filter

```
alight.filters.mylimit = function(exp, scope) {
    var ce = scope.$compile(exp);       // compile the input expression
    return function(value) {            // return handler
        var limit = Number(ce() || 5);
        return value.slice(0, limit)
    }
}
```

Example on jsfiddle

7.2 Input arguments

- expression - an input expression
- scope - current Scope

The filter should return a handler/function, one instance of filter.

7.3 alight.getFilter(name, scope, param)

This function look for a necessary filter, you can change it, for example you can make own location for your filters.

CHAPTER 8

Async filters

8.1 Overview

Async filters let you transform data in async mode, a filter should be placed in **alight.filters** or in scope.$ns for private filters.

Listing 8.1: Example of async filter

```
alight.filters.trottle = function(delay, scope, env) {
    delay = Number(delay);
    var to;
    return {
        onChange: function(value) {
            if(to) clearTimeout(to);
            to = setTimeout(function() {
                to = null;
                env.setValue(value);
                scope.$scan();
            }, delay);
        }
    }
}
```

8.2 Input arguments

- expression - an input expression
- scope - current Scope
- env - extra functional
- env.setValue(value) - set value of filter

Listing 8.2: Example of async filter

```
alight.filters.asyncFilter = function(expression, scope, env) {
    return {
        watchMode: 'deep',
        onChange: function(value) {},
        onStop: function() {}
    }
}
```

- watchMode, you can set 'simple'/'array'/'deep', if you need to change a watch mode for the input
- onChange - it's executed on every change of input
- onStop - it's executed when a watch object was removed

8.3 Examples

- Example trottle
- Example with loading from server

Text bindings

It’s a way to draw information to DOM:

```
<div>Hello {{name}}!</div>
<a href="http://example.com/{{link}}>link</a>
```

Also you can use method “bind once” for optimization, for that you should append “=” to begin of expression.

```
<div>Hello {{=name}}!</div>
<a href="http://example.com/{{=link}}>link</a>
```

Also you can use one time binding

If you can’t use tags `{{ }}`, you can change this to `{# #}`, `{< >}`, `## ##` or something like this, length of tags should be equal 2 symbols:

```
alight.utilits.pars_start_tag = '{#';
alight.utilits.pars_finish_tag = '#}';
```

For complex text bindings you can use text directives

CHAPTER 10

Text directives

10.1 Overview

An able to control a declarative data binding in the HTML

Listing 10.1: example how to use text directives

```
<div al-app>
    counter {{#counter}}
</div>
```

Listing 10.2: text directive counter

```
alight.text.counter = function(callback, expression, scope) {
    var n = 0;
    setInterval(function(){
        n++;
        callback(n)  // set result
        scope.$scan()  // $digest
    }, 1000);
}
```

10.2 Input arguments

- **callback** - a function to set a value
- **expression** - expression of directive
- **scope**
- env. **finally** - a function to set the final value, after that $watch will be removed.
- env.**setter** = callback

10.3 Examples

- Information output delay
- An counter
- Sample with bindonce

CHAPTER 11

One-time binding

11.1 Overview

Waits when an expression has a value (a non-undefined value), then stops watching. You need append ”::” in front of expression for using One-Time binding. It works with $watch, $watchText, directives and declarative bindings.

Listing 11.1: example

```
<div class="red {{::class}}"> {{::text}} </div>
<div al-show="::visible"></div>
<li al-repeat="it in ::list">...</li>
and so on
```

11.2 Examples

- Sample with declarative bindings
- Sample with al-repeat

CHAPTER 12

Isolated Angular Light

Angular Light can be invisible/inaccessible for alien code, you should make a function **alightInitCall-back(alightBuilder)**, it has to be callable for Angular Light. Angular Light starts the function and gives "alight-Builder()" as an argument on load.

It's useful feature, when your webapp will be used on alien page. It lets use different versions of Angular Light on the same page.

- Example 1
- Example 2

CHAPTER 13

Namespaces

Every scope and child scopes of it can have own sets of directives, controllers and filters. You should make an object **$ns** in scope. This can resolve conflict of names. This lets create private directives, filters and controllers.

If you want to inherit global directives, you may set **scope.$ns.inheritGlobal** = true.

Example

CHAPTER 14

A few ways to bind a model to the DOM

14.1 1. Manual binding, applyBindings

Listing 14.1: html

```
<div id="app">
    <input al-value="title" type="text" class="form-control" />
    <p>{{title}}</p>
</div>
```

Make scope, then to bind it to the DOM with **alight.applyBindings**

Listing 14.2: code

```
var tag = document.querySelector('#app');  // take the tag

var scope = alight.Scope();  // make a Scope
scope.title = 'Hello!';  // set init value

alight.applyBindings(scope, tag);  // apply bindings
```

Example on jsfiddle

14.2 2. Auto binding, al-app

alight.bootstrap is called on start system, it takes each element with **al-app** and execute **alight.applyBindings**

Listing 14.3: html

```
<div al-app al-init="title='Hello!'">
    <input al-value="title" type="text" class="form-control" />
    <p>{{title}}</p>
</div>
```

Example on jsfiddle

14.3 3. Manual binding with alight.bootstrap

You can bind custom elements with **alight.bootstrap**

Example on jsfiddle

Listing 14.4: html

```
<div id="app" al-init="title='Hello!'">
    <input al-value="title" type="text" class="form-control" />
    <p>{{title}}</p>
</div>
```

Listing 14.5: javascript

```
var tag = document.querySelector('#app');  // take the tag

alight.bootstrap([tag]);  // bind to DOM
```

14.4 4. Deferred binding, alight.bootstrap

Listing 14.6: html

```
<div al-app al-init="title='Hello!'">
    <input al-value="title" type="text" class="form-control" />
    <p>{{title}}</p>
</div>
```

Example on jsfiddle

14.5 5. To bind to element with no DOM

Example on jsfiddle

14.6 6. Manual binding #2

Example on jsfiddle

Listing 14.7: javascript

```javascript
alight.autostart = false;
setTimeout(alight.bootstrap, 2000);
```

Listing 14.8: html

```html
<div id="app"></div>
```

Listing 14.9: javascript

```javascript
var tag = document.createElement('div');  // make an element
// set up template
tag.innerHTML = '<input al-value="title" type="text" class="form-control" /><p>{{title}}</p>';
var scope = alight.Scope();  // make a scope
scope.title = 'Hello!';  // set init value

alight.applyBindings(scope, tag);  // apply bindings

document.querySelector('#app').appendChild(tag);  // append to DOM
```

Listing 14.10: html

```html
<div id="app">
    <input al-value="data.name" type="text" />
    {{data.name}} <br/>
    <button al-click="click()">Set Hello</button>
</div>
```

Listing 14.11: javascript

```javascript
alight.bootstrap({
    $el: '#app',
    data: {
        name: 'Some text'
    },
    click: function() {
        this.data.name = 'Hello'
    }
});
```

CHAPTER 15

Directive preprocessor

Directive preprocessor lets you control process of creating directives. You can make custom attributes and make different transformations.

Objects:

- **alight.directivePreprocessor** - a default preprocessor, you can change it
- **alight.directivePreprocessor.ext** - a list of handlers, you can append/remove them

Listing 15.1: Example how to create attribute 'bold'

```
alight.directivePreprocessor.ext.splice(1, 0, {
    code: 'bold',  // not necessary
    fn: function() {
        if(this.directive.bold) this.element.innerHTML = '<b>' + this.element.innerHTML + '</b>'
    }
})
```

Listing 15.2: How ot use it

```
alight.directives.al.example = {
    bold: true
}
```

- Example on plunkr
- Article [ru]: «»

CHAPTER 16

FAQ

16.1 How can I take a scope?

- Have a scope by tags
- Publish a scope from controller
- Publish a scope using a directive
- Take a scope by a name of controller
- A way to take Scope by element - alight.getScopeByElement(element)
- A way to take Scope by element with a directive
- How to call Alert from al-click
- How to simplify $parent.$parent.$parent...
- An one scope for a few applications
- **Where can I take a debug version** Here you can get release and debug of any version, also you can use bower: `bower install alight`
- **Where is $http service** Angular Light doesn't have it, you can use jQuery.ajax or anyone that you usually use. But there is `alight.f$.ajax` for inner purpose.
- **Angular Light updates a whole DOM or it tracks changes and update only changed elements?** Only changed elements.
- **I need Angular Light runs my function when a bindings process will finish.** You can observe it `scope.$scan("$finishBinding", callback)`
- **Is it possible to pass jquery $element to function from al-click? Something like al-click="someFunc($this)"** No, but you can extend the directive, or make a thin directive for this, example
- **How can I use al-repeat not as attribute, but as a comment?** al-repeat supports comment binding, example
- **Why al-show with an empty array doesn't hide my element: `al-show="model.array"`?** Because there is javascript and !!model.array always give you true, you can use `al-show="model.array.length"`
- **Where is "$odd, $even, $first, $last, $rest" for al-repeat?** You can append any attributes, example hot to append $odd $even
- **How to sort an array?** Manual sort Filter orderBy Sort props of an object Call a method
- **Where is al-select directive?** Here a few examples with select control.

- **Can I rename directives?** Yes, `alight.directives.al.myKeypress = alight.directives.al.keypress`
- **How to call my function when Scope.$scan finish digets process?** You can pass your function as callback `Scope.$scan(callback)`
- **How to redraw bind-once?** Example

Examples