alight Documentation

Release 0.11.0

Oleg Nechaev

November 29, 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.bootstrap(option)

It lets you a few ways to bind DOM, read more

1.3 alight.bind(ChangeDetector, element, option)

Bind a change detector to the DOM element, alias **alight.applyBindings**

- ChangeDetector - an instance of ChangeDetector
- 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 cd = alight.ChangeDetector();
var element = document.body;
alight.bind(cd, element);
```

1.4 alight.ChangeDetector([scope])

Create a new change detector, read more

1.5 alight.ctrl

Dictionary of controllers, alias for **alight.directives.ctrl**

1.6 alight.filters

Dictionary of filters

1.7 alight.directives

Dictionary of directives, short alias **alight.d**

1.8 alight.text

Collection of text directives

1.9 alight.hook

Different hooks

1.10 alight.nextTick(callback)

Execute the function on next tick

CHAPTER 2

alight.bootstrap

It lets you a few ways to bind DOM.

2.1 Binding to all elements with attribute [al-app]

- alight.bootstrap()

2.2 Binding by selector

- alight.bootstrap('#someId')
- alight.bootstrap('#someId', scope)

2.3 Binding to element

- alight.bootstrap(element)
- alight.bootstrap(element, scope)

2.4 Binding to array of elements

- alight.bootstrap([element1, element2, element3])
- alight.bootstrap([element1, element2, element3], scope)

Note: If you provide **scope**, then all elements will be binded to this scope, otherwise every dom will have own scope.

2.5 Example

Listing 2.1: Example of bootstrap

```
alight.bootstrap('#app', {
    name: 'world',
    click: function() {
        this.name = 'user'
    }
});
```

CHAPTER 3

Change Detector

It lets you observe changes in your scope

3.1 ChangeDetector.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
- **"$finishScanOnce"**
- **"$onScanOnce"** - the callback is called in scan loop

Option:

- **option** = true or **option.isArray** = true - watch an array
- **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)

3.2 ChangeDetector.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 3.1: Example of $compile

```
var scope = {};
var cd = alight.ChangeDetector(scope)
var fn = cd.compile('"hello " + title')
scope.title = 'linux'
fn(scope) // return "hello linux"
scope.title = 'macos'
fn(scope) // return "hello macos"
```

Listing 3.2: Example with input

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

Listing 3.3: Example with no_return

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

3.3 ChangeDetector.eval(expression)

Execute an expression

3.4 ChangeDetector.watchText(tpl, callback)

Track the template

3.5 ChangeDetector.new([scope])

Create a child ChangeDetector, if scope is omitted, then it will used parent scope

3.6 ChangeDetector.destroy()

Destroy the Scope.

3.7 ChangeDetector.scan(callback or option)

Starts the search for changes, returns a watch statistic

- **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.skipWatch** - skip specific watch
- **option.late =** *(true/false)* - If there is a few $scan commands, Angular Light will call only last one.

Listing 3.4: Example with $scan

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

3.8 ChangeDetector.getValue(name)

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

3.9 ChangeDetector.setValue(name, value)

Set the value of the variable

Listing 3.5: Example with $setValue

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

// equal
var scope = {}
var cd = alight.ChangeDetector(scope);

cd.setValue('var', 1);
cd.setValue('path.var', 2);
cd.setValue('path[key]', 3);
```

CHAPTER 4

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 4.1: Example of directive al-text

```
alight.directives.al.text = function(scope, cd, element, name, env) {
    // Track to the variable
    cd.watch(name, function(text) {
        // set a text to the DOM-element
        $(element).text(value)
    });
};
```

Input arguments:

- **scope** - current Scope
- **cd** - change detector
- **element** - element of DOM
- **name** - value of attribute
- **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
- env. **stopBinding** = false - stop binding for child elements

4.1 Attributes of directive:

- **priority** - you can set priority for a directive
- **template** - custom template
- **templateUrl** - link to template
- **scope** (false/true) - if it is true, there will be a new, empty scope
- **ChangeDetector** (false/true/'root') - create a new change detector
- **restrict** = 'A', can be 'AEM', 'A' matches attribute name, 'E' matches element name, 'M' matches class name
- **link** - the method is called after template, scope
- **init** - the method is called when the directive is made, before template, scope. You usually need **link** instead of it.
- **stopBinding** (true) - stop binding for child elements
- *anyAttr* - you can make a custom attribute, look. directive preprocessor

Listing 4.2: Directives with attributes

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

If "stopBinding" is true, then the process of binding will skip 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 5

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 5.1: Inherit al-value

```
alight.directives.al.valueDelay = function(scope, cd, element, value, env) {
    // create a source directive
    var dir = alight.directives.al.value(scope, cd, element, value, env);

    // 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;
}
```

5.1 Examples of inheritance

Examples below for v0.10, but in v0.11 it works similar * 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 6

Directives

6.1 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

6.2 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

6.3 Special directives

- al-app, init application with current element, examples
- al-cloak, hide current element until activate the application, examples
- al-class/al-css, todo sample, animated sample
- **al-style** examples
- **al-show**, sample with animation
- 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-src
- al-stop, stops a bind process for the element and his children.
- al-text, example
- al-select + al-option, example

6.4 Bind-once

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

CHAPTER 7

Base filters

7.1 date

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

Listing 7.1: example

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

7.2 filter

- To filter the list
- Input argument: variable

Example

7.3 slice

- To slice the list
- input arguments: numbers

Listing 7.2: example

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

7.4 generator

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

Example

Listing 7.3: example

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

7.5 toArray

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

Listing 7.4: example

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

Example

7.6 orderBy

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

Listing 7.5: example

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

Example

7.7 throttle

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

Example

7.8 json

- display data in json style

Listing 7.6: example

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

Listing 7.7: example

```
{{data.value | json}}
{{this | json}}
```

CHAPTER 8

Create filters

8.1 Overview

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

Listing 8.1: Example filter

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

Example on jsfiddle

8.2 Input arguments

- expression - an input expression
- cd - change detector

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

CHAPTER 9

Async filters

9.1 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 9.1: Example of async filter

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

9.2 Input arguments

- expression - an input expression
- cd - change detector
- env - extra functional
- env.setValue(value) - set value of filter

Listing 9.2: Example of async filter

```
alight.filters.asyncFilter = function(expression, cd, 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

9.3 Examples

- Example trottle
- Example with loading from server

CHAPTER 10

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

For complex text bindings you can use text directives

CHAPTER 11

Text directives

11.1 Overview

An able to control a declarative data binding in the HTML

Listing 11.1: example how to use text directives

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

Listing 11.2: text directive counter

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

11.2 Input arguments

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

11.3 Examples

- Information output delay
- An counter
- Sample with bindonce

CHAPTER 12

One-time binding

12.1 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 12.1: example

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

12.2 Examples

- Sample with declarative bindings
- Sample with al-repeat

CHAPTER 13

Isolated Angular Light

Angular Light can be invisible/inaccessible for alien code, you should make a function **alightInitCallback(alightBuilder)**, it has to be callable for Angular Light. Angular Light starts the function and gives "alightBuilder()" 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 14

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 15

A few ways to bind a model to the DOM

15.1 1. Manual binding

Listing 15.1: html

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

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

Listing 15.2: code

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

var scope = {
    title: 'Hello!'
};
var cd = alight.ChangeDetector(scope);

alight.bind(cd, tag);  // apply bindings
```

Example on jsfiddle

15.2 2. Auto binding, al-app

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

Listing 15.3: html

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

Example on jsfiddle

15.3 3. Manual binding with alight.bootstrap

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

Listing 15.4: html

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

Listing 15.5: javascript

```
alight.bootstrap('#app');  // bind to DOM
```

Example on jsfiddle

15.4 4. To bind to element with no DOM

Listing 15.6: html

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

Example on jsfiddle

15.5 5. Manual binding #2

Example on jsfiddle

Listing 15.7: 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>';

alight.bootstrap(tag, {
    title: 'Hello!'
})

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

Listing 15.8: html

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

Listing 15.9: javascript

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

CHAPTER 16

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 16.1: 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 16.2: How ot use it

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

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

CHAPTER 17

Migration from 0.10 to 0.11

17.1 Controllers

Listing 17.1: html 0.10

```
<div al-controller="main"></div>
```

Listing 17.2: js 0.10

```
alight.controllers.main = function(scope) {};
```

Listing 17.3: html 0.11

```
<div ctrl-main></div>
```

17.2 Directives

17.3 Scope

Scope - it was a mix of change detector and user's data, in v0.11 Scope was devided to ChangeDetector and scope={} (user's data), it gives different advatages.

Listing 17.4: js 0.11

```
alight.ctrl.main = function(scope, cd) {};
```

Listing 17.5: with "isolated" scope 0.11

```
alight.ctrl.main = {
    scope: true,
    link: function(scope, cd) {
        scope.$parent // parent scope
    }
};
```

Listing 17.6: directive in 0.10

```
alight.directives.al.test = function(element, value, scope, env) {};
```

Listing 17.7: directive in 0.10

```
alight.directives.al.test = {
    scope: true,  // true / 'isolate' / 'root'
    link: function(element, value, scope, env) {}
}
```

Listing 17.8: directive in 0.11

```
alight.directives.al.test = function(scope, cd, element, value, env) {};
```

Listing 17.9: directive 0.11

```
alight.directives.al.test = {
    scope: true,  // it make a new clean scope (object)
    ChangeDetector: true // true / 'root'
    link: function(scope, cd, element, value, env) {}
}
```

Listing 17.10: Scope 0.10

```
scope = alight.Scope()
scope.name = 'linux'

scope.$watch('name', function(value) {
}, {
    init: true  // it calls the callback immediately
})

scope.$scan()
scope.$destroy()
```

Listing 17.11: v0.11

```
scope = {
    name: 'linux'
}
cd = alight.ChangeDetector scope
// now cd.scope === scope

cd.watch('name', function(value) {});
// you don't need set 'init' anymore, the callback is executed on finish binding process, also you ca

cd.scan()
cd.destroy()
```

CHAPTER 18

FAQ

18.1 How can I take a scope?

for Angular Light v0.10 and older

- 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
- **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?** You can use $event that contains element, al-click="someFunc($event)"
- **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 how to append $odd $even
- **How to sort an array?** Manual sort Filter orderBy Sort props of an object Call a method
- **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