AngularJS

陳葵懋 Ian Chen
http://codeian.idv.tw/

codeian.idv.tw

- Microsoft MVP
- TechDays Taiwan 2014 / 2015講師
- 2015 微軟實戰課程日講師
- Channe 9線上課程講師
- K.NET & Study4.TW社群講師
- HTML5 & JavaScript 程式開發實戰書籍共同作者
- Microsoft MSDN 技術文章作者

Agenda

- AngularJS基礎概念
- AngularJS Directive與Controller
- AngularJS表單與資料繫結
- Angularjs Http Client呼叫後端API
- AngularJS ngRoute / ui-routing
- AngularJS directive & component

AngularJS基礎概念

- 前端 Javascript 框架 by Google
- SPA式應用程式的框架
- 基於"資料"操控，而非DOM的操控
  - Model、View、Controller
  - Model、View、Controller
  - Model、View、Controller
  - 很重要所以要說三次

Why AngularJS

- 輸出使用者文字

jQuery

```
<script>
    $(document).ready(function () {
        $("#myname").keydown(function () {
            $('#viewlab').text($('#myname').val());
        });
        $("#myname").keyup(function () {
            $('#viewlab').text($('#myname').val());
        });
    });
</script>
<input id="myname" type="text"/>
<label id="viewlab"></label>
```

Why AngularJS

- 輸出使用者文字

AngularJs

```
<body ng-app>
    <input type="text" ng-model="name" />
    <label ng-bind="name"></label>
</body>
</html>
```

Why AngularJS

- 輸出使用者文字

```
<body ng-app>
    <input type="text" ng-model="name" />
    <label ng-bind="name"></label>
</body>
</html>
```

AngularJs

程式碼勒？

Why AngularJS

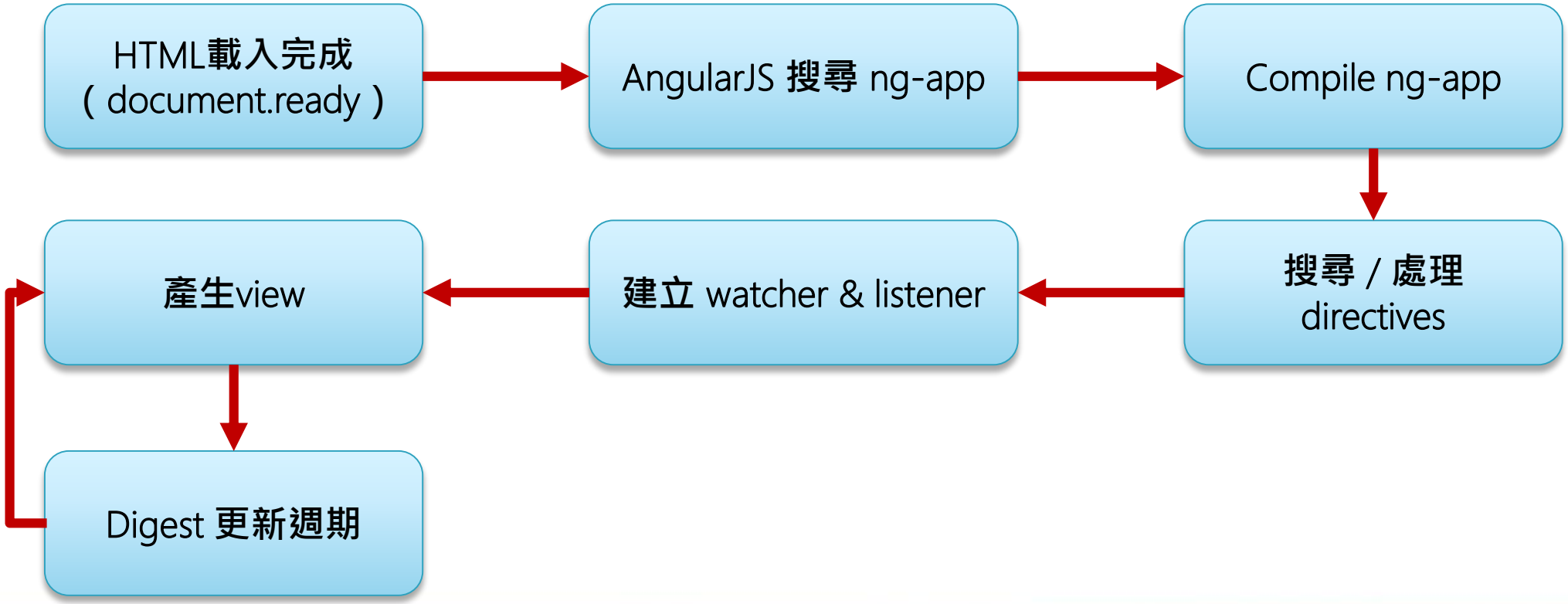
- 輸出使用者文字

```
<body ng-app>
    <input type="text" ng-model="name" />
    <label ng-bind="name"></label>
</body>
</html>
```

AngularJs

AngularJs Magic

AngularJS 生命週期



AngularJS 基礎概念

```
<head>
    <title></title>
    <meta charset="utf-8" />
    <script src="Scripts/angular.min.js"></script>
</head>
<body ng-app>
    <input type="text" ng-model="name" />
    <label ng-bind="name"></label>
</body>
```

AngularJS 基礎概念

```
<head>
    <title></title>
    <meta charset="utf-8" />
    <script src="Scripts/angular.min.js"></script>
</head>
<body ng-app>
    <input type="text" ng-model="name" />
    <label ng-bind="name"></label>
</body>
```

AngularJS 基礎概念

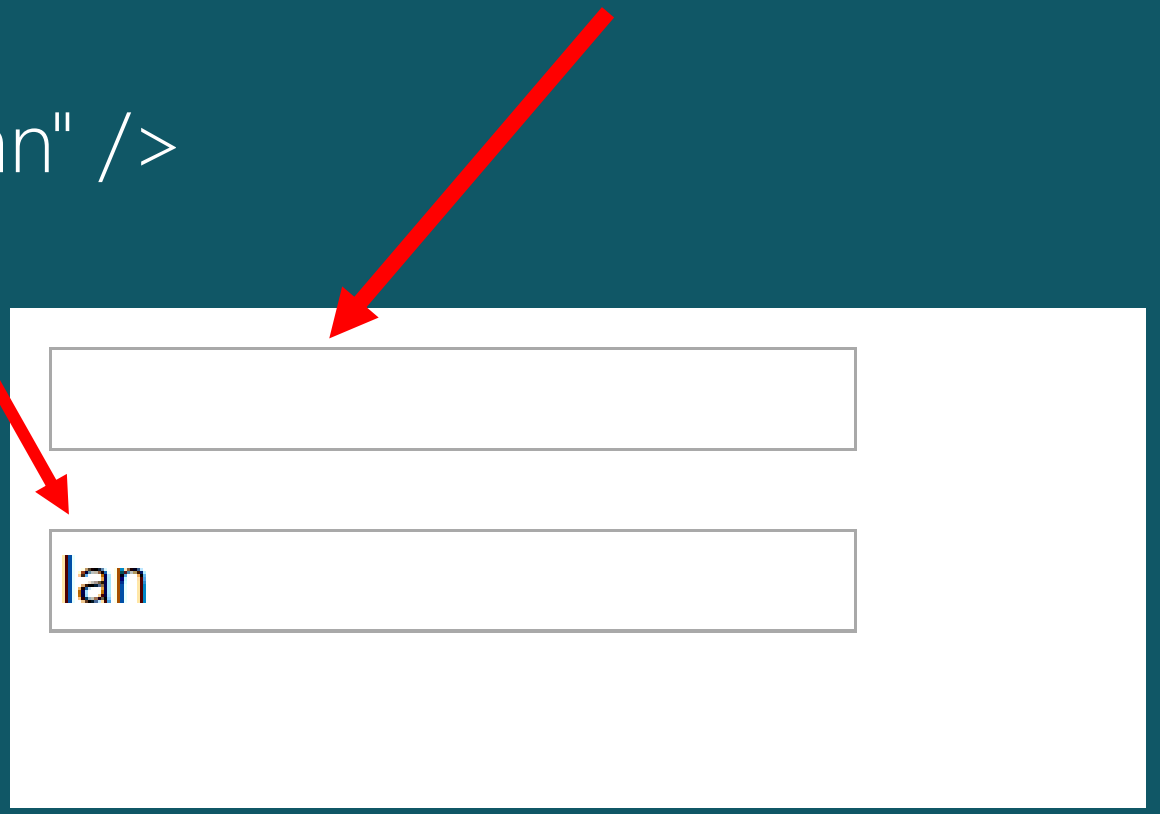
```
<head>
    <title></title>
    <meta charset="utf-8" />
    <script src="Scripts/angular.min.js"></script>
</head>
<body ng-app>
    <input type="text" ng-model="name" />
    <label ng-bind="name"></label>
</body>
```

AngularJS 基礎概念

```
<head>
    <title></title>
    <meta charset="utf-8" />
    <script src="Scripts/angular.min.js"></script>
</head>
<body ng-app>
    <input type="text" ng-model="name" />
    <label ng-bind="name"></label>
</body>
```

AngularJS 基礎概念

```
<body ng-app>
    <input type="text" ng-model="name" value="Ian" />
    <p/>
    <input type="text" value="Ian" />
</body>
```



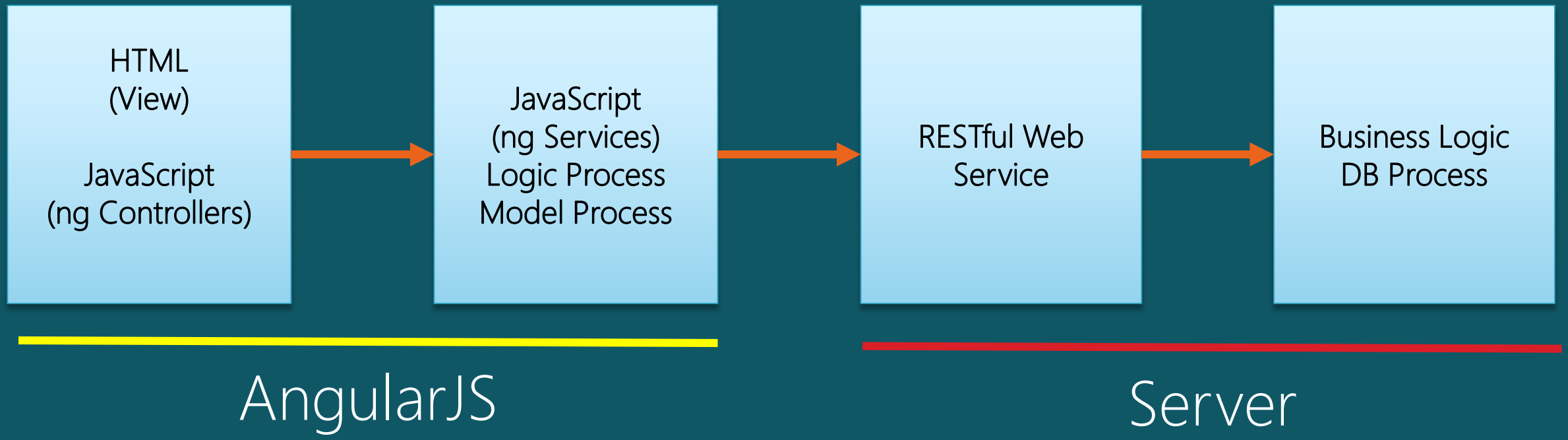
AngularJS 基礎概念

- 以model為概念，以model的思維出發

```
<body ng-app="myapp">
    <div ng-controller="mainctrl as ctrl">
        <input type="text" ng-model="ctrl.name" />
    </div>
    <script>
        var myapp = angular.module('myapp', []);
        myapp.controller('mainctrl', [function () {
            var self = this;
            self.name = 'Ian';
        }]);

    </script>
</body>
```

AngularJS 基礎概念



AngularJS Directive與Controller

- ng-app
  - 宣告是個AngularJS app
  - 一個HTML Page可以有多個宣告
  - Any HTML element
- <htmlElment ng-app="myapp" >

AngularJS Directive與Controller

- Hello AngularJS

```
<body ng-app>
    <input type="text" ng-model="name" />
    Hello:<span ng-bind="name"></span>
    <script src="Scripts/angular.min.js"></script>
</body>
```

AngularJS Directive與Controller

- Hello AngularJS

```
<body ng-app="myapp">
    <input type="text" ng-model="name" />
    Hello:<span ng-bind="name"></span>
    <script src="Scripts/angular.min.js"></script>
    <script>
        var app = angular.module('myapp', [ ]);
    </script>
</body>
```

AngularJS Directive與Controller

- angular.module
  - 模組化
  - 有效控制相關成員

```
<body ng-app="myapp">
    <!-- ..... -->
    <script>
        var app = angular.module('myapp', []);
    </script>
</body>
```

AngularJS Directive與Controller

- angular.module(‘myapp’, [ ])
  - 宣告一個新的module
  - myapp – module name
  - [ ] – 指定其它相依的module
    - [‘aaa’,’bbb’]
- angular.module('myapp'~~, [ ]~~);
  - 直接取用其它檔案已定義的 module

AngularJS Directive與Controller

- Controller
  - view 與 model 間的溝通通道
  - 非UI的Logic應由Services負責

```
<body ng-app="myapp">
    <div ng-controller="mainctrl as ctrl">
        <input type="text" ng-model="ctrl.name" />
    </div>
    <script src="Scripts/angular.min.js"></script>
    <script>
        var app = angular.module('myapp', []);
        app.controller('mainctrl', [function () {
            var self = this;
            self.name = 'ian';
        }]);

    </script>
</body>
```

AngularJS Directive與Controller

- ng-init

```
<body ng-app="myapp">
    <div ng-controller="mainctrl as ctrl" >
        <input type="text" ng-model="ctrl.name" />
        <input type="text" ng-model="ctrl.name2"
               ng-init="ctrl.name2='ian2'"  />
    </div>
    <script>
        var app = angular.module('myapp', []);
        app.controller('mainctrl', [
            function () {
                var self = this;
                self.name = 'ian1';
            }
        ]);
    </script>
</body>
```

AngularJS Directive與Controller

- ng-click

```
<button ng-click="ctrl.calbmi()">ADD</button>
```

```
var app = angular.module('bmiapp', [ ]);
app.controller('bmictrl', [
    function () {
        var self = this;
        self.calbmi = function ( ) {
            //.....
        };
} ]);
```

AngularJS Directive與Controller

```
app.controller ( 'mainctrl' , function ( ) {
        /*....*/
    } );
```

建議語法

```
app.controller ( 'mainctrl' , [ function ( ) {
        var self = this;
        self.name = 'ian';
    } ] );
```

安全式相依注入

```
app.controller('mainctrl', function ($http) { });
app.controller('mainctrl', ['$http', function ($http) { }]);
```


```
app.controller('mainctrl', function (abc) { }); //run error
app.controller('mainctrl', ['$http', function (abc) { }]);
```

$scope 與 controller as

- $scope
  - AngularJS 1.2 之前
  - $scope.變數名稱
- controller as
  - 使用 this 定義 controller 實體裡的變數成員
  - controller與變數成員關係明確
  - 將 this 以區域變數代理 #令人搞不清的 javascript this 現象

$scope 與 controller as

```
<body ng-app="myapp">
    <div ng-controller="mainctrl as ctrl">
        <input type="text" ng-model="ctrl.name" />
    </div>
    <script src="Scripts/angular.min.js"></script>
    <script>
        var app = angular.module('myapp', [ ]);
        app.controller('mainctrl', [ function ( ) {
            var self = this;
            self.name = 'ian';
        }]);
    </script>
</body>
```

$scope 與 controller as

```
<div ng-controller="ParentCtrl">
    ParentController: <input type="text" ng-model="empname" />
    <div ng-controller="ChildCtrl">
        ChildController: <input type="text" ng-model="empname" />
    </div>
</div>
```

```
<script>
        var app = angular.module('myapp', []);
        app.controller('ParentCtrl', function ($scope) {
            $scope.empname = "ian";
        })
        .controller('ChildCtrl', function ($scope) {
            alert($scope.empname);
        });
    </script>
```

$scope 與 controller as

```
<div ng-controller="ParentCtrl">
    ParentController: <input type="text" ng-model="p.empname" />
    <div ng-controller="ChildCtrl">
        ChildController: <input type="text" ng-model="c.empname" />
    </div>
</div>
```

```
<script>
    var app = angular.module('myapp', []);
    app.controller('ParentCtrl', function ($scope) {
        $scope.p = {
            empname: "ian"
        };
    })
    .controller('ChildCtrl', function ($scope) {
    });
</script>
```

$scope 與 controller as

```
<div ng-controller="ParentCtrl as ctrlp">
    ParentController: <input type="text" ng-model="ctrlp.empname" />
    <div ng-controller="ChildCtrl as ctrlc">
        ChildController: <input type="text" ng-model="ctrlc.empname" />
    </div>
</div>
```

```
<script>
    var app = angular.module('myapp', []);
    app.controller('ParentCtrl', function () {
        var self = this;
        self.empname = 'ian';
    })
    .controller('ChildCtrl', function () {
        var self = this; alert(self.empname);
    });
</script>
```

Lab
建立BMI試算網頁
BMI公式=體重（公斤）/ 身高（公尺）的平方

AngularJS表單與資料繫結

- controller 不應該有 jQuery selector
- controller 不應該有 document.getelementbyid
- controller 針對 model 操作

AngularJS表單與資料繫結

- ng-model
  - two-way data binding
  - <input type = "text" ng-model = "ctrl.bmival"/>
  - ~~~~
- ng-bind
  - one-way data binding
  -

AngularJS表單與資料繫結

- ng-bind
  - 與 {{ }} 作用相同
  - {{ }} 效能較差，UI 可能發生{{ }} 殘影

```
Hello:<span ng-bind="name"></span>

Hello:<span>{{name}}</span>
```

AngularJS表單與資料繫結

- ng-click
  - 使用者點擊button才有效
- ng-submit
  - enter 、點擊button

AngularJS表單與資料繫結

ng-model 會自動建立 emp object

```
<div ng-controller="mainctrl as ctrl">
    <form ng-submit="ctrl.send()">
        <input type="text" ng-model="ctrl.emp.name" />
        <input type="text" ng-model="ctrl.emp.no" />
        <input type="submit" value="send" />
    </form>
</div>
<script>
    var app = angular.module('myapp', []);
    app.controller('mainctrl', [
        function () {
            var self = this;
            self.send = function () {
                console.log('data send:' + self.emp);
            };
        }
    ]);
```

AngularJS表單與資料繫結

- $invalid 驗證
  - 具名form
  - 搭 HTML 5 驗證屬性 required
  - AngularJS 驗證器
  - 任一驗證無效時，其值為true
- $valid
  - 所有驗證通過，其值為true
- 語法：**表單名稱.欄位名稱.驗證屬性**

AngularJS表單與資料繫結

- ng-pattern
  - 建立樣式驗證器
  - ng-pattern="/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,12}$/"

```
ID : <input type="text" name="userid" ng-model="ctrl.user.uid"
           required
           ng-pattern="/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,12}$/"/>
<span style="color:red" ng-show="empform.userid.$error.required">*</span>
<span style="color:red" ng-show="empform.userid.$error.pattern">
必須包含大寫、小寫英文字母及數字，且長度為6~12碼
</span>
```

AngularJS表單與資料繫結

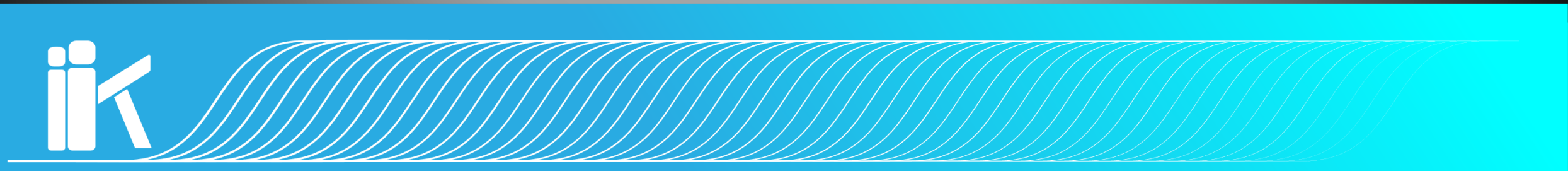
- $error
  - 表單錯誤資訊物件
  - empform.empname.$error

```
ength="5" />
 ng-show="empform.empno.$error.required">*</span>
 ng-show="empform.empno.$error.minlength">工號最小長度5碼</span>

lue="send" ng-disabled="empform.$invalid" />
```

Lab

- 請建立以下驗證並顯示提示訊息
- Email 格式
- Email 必填
- 帳號必填且長度至少6碼



AngularJS表單與資料繫結

- Checkbox 元素
  - 雙向綁定 ng-model
  - 單向綁定 ng-checked

```
ng-model : <input type="checkbox" ng-model="ctrl.ck" />
ng-model2 : <input type="checkbox"
                   ng-model="ctrl.ck2"
                   ng-true-value="'Y'"
                   ng-false-value="'N'" />
ng-checked : <input type="checkbox" ng-checked="ctrl.ck2==='Y'" />
```

AngularJS表單與資料繫結

- radio元素
  - ng-model + value

```
male : <input type="radio" name="gender"
           ng-model="ctrl.gender"
           value="male" />
female : <input type="radio" name="gender"
           ng-model="ctrl.gender"
           value="female" />
```

AngularJS表單與資料繫結

- select 元素
  - ng-options + ng-model
  - ng-options=" AngularJS Expression for variable in array "

```
<div ng-controller="mainctrl as ctrl">
    <select ng-model="ctrl.emptype"
            ng-options="types.type for types in ctrl.emptypesmodel">
        <option value="">-- 請選擇 --</option>
    </select>
    你選擇的是:<span ng-bind="ctrl.emptype.type"></span>
</div>
```

AngularJS表單與資料繫結

- ng-repeat
  - for each
  - data view template
  - variable in array expression
  - (key , value) in objects expression

AngularJS表單與資料繫結

```
<div ng-repeat="emp in ctrl.emps">
        <span ng-bind="emp.no"></span>
        <span ng-bind="emp.name"></span>
</div>
```

```
var self = this;
self.emps = [
        {no:'001',name:'ian'},
        {no:'002',name:'andy'},
        {no:'003',name:'steve'},
];
```

AngularJS表單與資料繫結

```
<div ng-repeat="(name,emp) in ctrl.emps">
        <span ng-bind="emp.no"></span>
        <span ng-bind="name"></span>
        <span ng-bind="emp.enterdate"></span>
</div>
```

```
var self = this;
self.emps = {
        ian: { no: '001', enterdate: '2001/1/7' },
        andy: { no: '002', enterdate: '2001/5/7' },
        steve: { no: '003', enterdate: '2002/11/13' }
};
```

AngularJS表單與資料繫結

- ng-repeat helper
  - $first - 第一筆(bool)
  - $middle –中間資料(bool)
  - $last – 最後一筆(bool)
  - $index – 資料列index
  - $even – 奇數列(bool)
  - $odd – 偶數列(bool)

Lab

- 利用 HTML Table tag + ng-repeat 建立資料List顯示頁面
- 必須顯示資料筆數號

Dependency Injection System

- 相依性注入(DI)
- 將常用、重複性使用的功能包裝成元件
- 寬鬆耦合
  - 實體非由caller直接建立(new Instance) - 高耦合

想像一件事
透過某個網路服務取得資料

Dependency Injection System

- 非DI
  - 可能在每一個 controller 撰寫
  - 抽換web api url ?
  - 抽換服務類型 http

```
app.controller('thctrl', [function () {
        var self = this;
        this.emps = $http.get( ' web api url ' );
}]);
```

Dependency Injection System

- DI
  - 服務是由外面注入進來
  - 只管開口，不管實作
  - 服務底層可抽換，只要get方法名稱不變
  - 服務是由外面注入進來

```
app.controller('thctrl', ['empsservice',
        function (empswebapi) {
                var self = this;
                self.emps = empswebapi.get();
}]);
```

AngularJS DI – 相依性注入

- services name
  - $log , $http
- 變數
  - logservice , httpservice
- 依注入順序對應變數

```
app.controller('myctrl', ['$log', '$http'
        , function (logservice, httpservice) {
            var self = this;
            logservice.log('di');
}]);
```

services

- 共享狀態/行為的函式或物件
- 共用單一實體 – 實體化一次
- factory
- service
- provider
- 只是不同的語法糖

services

```
function factory(name, factoryFn, enforce) {
  return provider(name, {
    $get: enforce !== false ? enforceReturnValue(name, factoryFn) : factoryFn
  });
}

function service(name, constructor) {
  return factory(name, ['$injector', function($injector) {
    return $injector.instantiate(constructor);
  }]);
}
```

Services or Controlls

controlls	services
處理 view logic	處理 business logic
一次性使用	重複性使用
負責處理與使用者UI互動，UI處理等	負責與後端server連繫或整合其它第3方services
controlls內的成員隨controls建立 / 消滅	整個應用程式層級共用的狀態 / 行為

Services or Controlls

- 取得應用程式設定值
  - ?
- 驗證表單資料
  - ?
- 取得登入者資訊
  - ?

RESTful Web Service

- Representational State Transfer
- OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT

Action	RESTful Service	HTTP Method
新增	http://xxx.xxx.xxx/employees	POST
刪除	http://xxx.xxx.xxx/employees/1	DELETE
修改	http://xxx.xxx.xxx/employees/1	PUT
單一查詢	http://xxx.xxx.xxx/employees/1	GET
列表	http://xxx.xxx.xxx/employees/	GET

XMLHttpRequest

```
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      document.getElementById("demo").innerHTML = xhttp.responseText;
    }
  };
  xhttp.open("GET", "demo_get.asp", true);
  xhttp.send();
}
</script>
```

JavaScript Callback

- 一種設計模式
- 確保事件執行順序
  - A fun( )　　B fun( )　　C fun( )

Asynchronous XMLHttpRequest

```
function xhrSuccess () { this.callback.apply(this, this.arguments); }

function xhrError () { console.error(this.statusText); }

function loadFile (sURL, fCallback /*, argumentToPass1, argumentToPass2, etc. */) {
  var oReq = new XMLHttpRequest();
  oReq.callback = fCallback;
  oReq.arguments = Array.prototype.slice.call(arguments, 2);
  oReq.onload = xhrSuccess;
  oReq.onerror = xhrError;
  oReq.open("get", sURL, true);
  oReq.send(null);
}
```

https://developer.mozilla.org/zh-TW/docs/Web/API/XMLHttpRequest/Synchronous_and_Asynchronous_Requests

$http

- AngularJS XHR (XMLHttpRequest)
- Promise – 非同步

$http

- 沒有post資料
  - 參數1 : url ; 參數2 : 組態設定
  - get(url, [config]);
  - delete(url, [config]);
- 有post資料
  - 參數1 : url ; 參數2 : data; 參數3 : 組態設定
  - post(url, data, [config]);
  - put(url, data, [config]);

$http

- 每一個非同步會回傳一個Promise物件
- then函式具有success , error二個處理function
- 非同步作業完成後，會觸發then裡的success 或 error function
- Promise 機制讓程式邏輯不用處於停頓等待狀態

$http

```
$http.get('/someUrl', config).then(successCallback, errorCallback);
$http.post('/someUrl', data, config).then(successCallback, errorCallback);
```

```
$http.get('api url')
.then(
    function (response) {
        //處理回傳資料
    }
    , function (error) {
        //處理error
    });
```

$http

- then 可以再次回傳 promise 物件，串連多個then作業程序

```
$http.get('api url')
.then(function (response) {
    //.....
})
.then(function (response) {
    //....
}, function (error) {
    //處理error
});
```

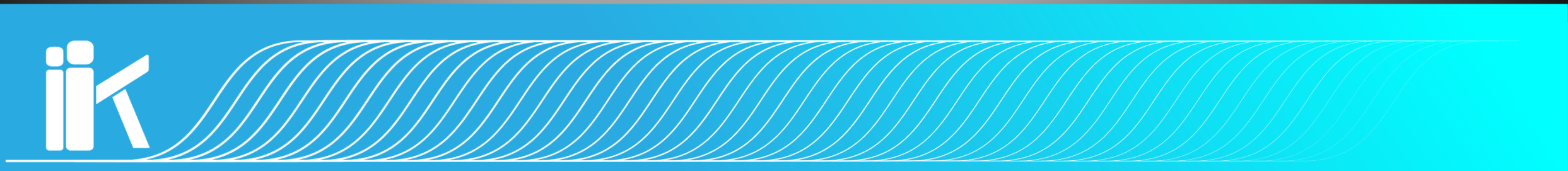
$http

- config
  - 有預設組態
  - https://docs.angularjs.org/api/ng/service/$http#usage
  - customize your config

```
$http.get('http://myngwebapi.azurewebsites.net/api/employee', {
    headers: { 'Authorization': 'Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==' }
});
```

Lab

- 利用 $http 呼叫 Web API 取得資料
- 將取得的資料以列表方式的UI呈現
- Web API URL : http://myngwebapi.azurewebsites.net/api/employee



Lab

- 以 factory 封裝 $http 呼叫 Web API
- 將取得的資料以列表方式的UI呈現
- factory 內容請獨立為一個 js 檔案
- Web API URL : http://myngwebapi.azurewebsites.net/api/employee

$resource service

- base on $http
- 更易於處理呼叫RESTful Service 的CRUD
- 獨立的AngularJS service
  - angular-resource.js

$resource service

- module 載入其它模組
  - angular.module('myapp', ['ngResource']);
- controller相依性注入
  - app.controller('mainctrl', ['$resource', function ($resource){..}]);

$resource service

- $resource(url, [paramDefaults], [actions]);
- url
  - web service url
- paramDefaults
  - 以json格式表示的url 參數值
  - 參數以 :+名稱表示，/employee/:id
  - { id: '001'} 表示定義預設值，
  - { id: '@id'}表示參數值由request 的 json object裡取得id屬性的值
- actions
  - 定義額外的http作業

```
var api = $resource(url + '/employee/:id'
    , { id: '@id' }
    , {
        update: {
            method: 'PUT'
        }
    });
```

$resource service

- 預設http action
- isArray
  - 接收回傳Array Data

```
{
  'get':    {method:'GET'},
  'save':   {method:'POST'},
  'query':  {method:'GET', isArray:true},
  'remove': {method:'DELETE'},
  'delete': {method:'DELETE'}
};
```

$resource service

- 自訂義http action

```
var api = $resource(url + '/employee/:no', { no: '@No' }
                , {
                    update: {
                        method: 'PUT'
                    },
                    add: {
                        method: 'POST', isArray: true
                    }
                });
```

HomeWork Lab

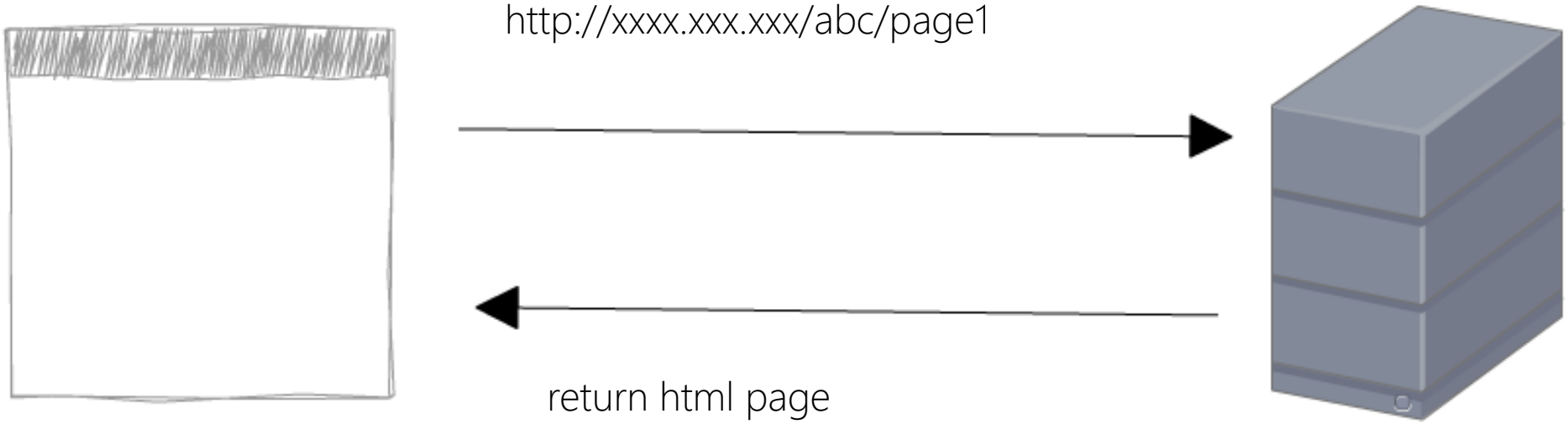
- 以 factory 封裝 $resource呼叫 Web API
- 將取得的資料以列表方式的UI呈現
- factory 內容請獨立為一個 js 檔案
- Web API URL : http://myngwebapi.azurewebsites.net/api/employee

Route路由

- 處理多個 view 在同一個SPA頁面
  - 不重新載入整個頁面
- ngRoute
  - AngularJS 1.2 ngRoute是option模組

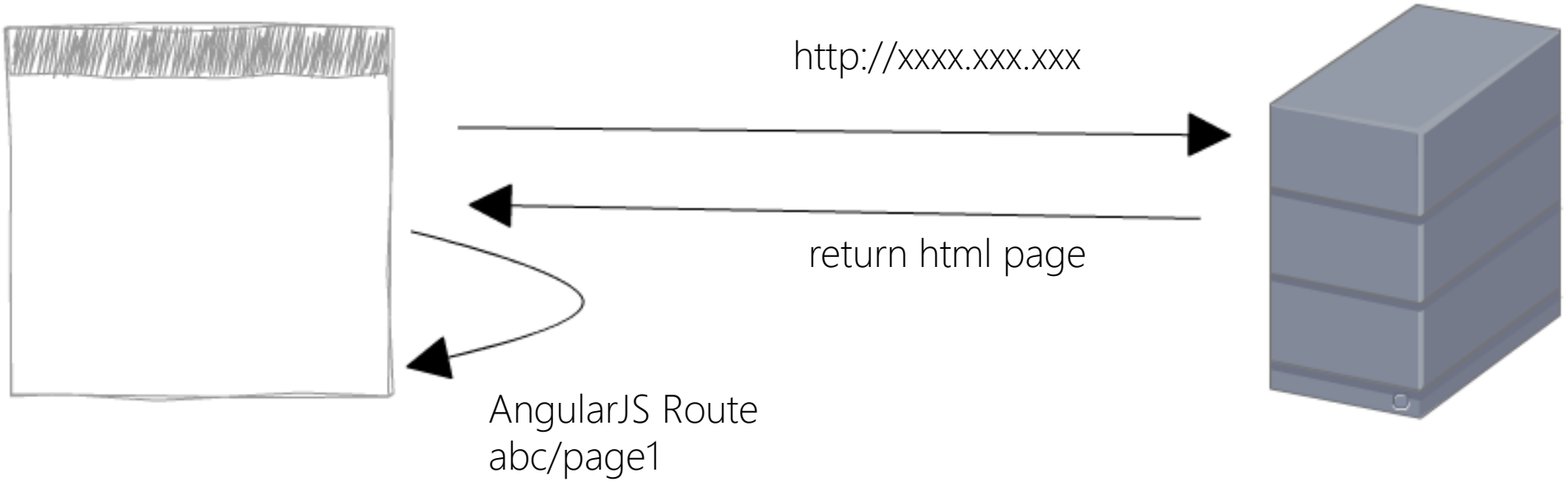
Route路由

http://xxxx.xxx.xxx/abc/page1



Route路由

http://xxxx.xxx.xxx/#/abc/page1



ngRoute

- include ngRoute JS
  - angular-route.min.js
- module 相依性注入
  - angular.module('myapp', ['ngRoute'])
- use ng-view directive
  - <div ng-view>

ngRoute

- config route
  - when
  - otherwise

```
app.config(['$routeProvider', function ($routeProvider) {
        $routeProvider.when('/', {
            redirectTo: '/'
        }).when('/page2', {
            template: '<h3>This is Page2</h3>'
        }).otherwise({
            redirectTo: '/'
        });
    }]);
```

ngRoute

- when

```
$routeProvider.when(
        url,
        {
            template: string,
            templateUrl: string,
            controller: string,function or array
            controllerAS: string,
            resolve:object<key,function>
        });
```

ngRoute

- 觸發組態檔的url

```
$routeProvider.when(
        url,
        {
            template: string,
            templateUrl: string,
            controller: string,function or array
            controllerAS: string,
            resolve:object<key,function>
        });
```

ngRoute

- Html string直接插入ng-view

```
$routeProvider.when(
        url,
        {
            template: string,
            templateUrl: string,
            controller: string,function or array
            controllerAS: string,
            resolve:object<key,function>
        });
```

ngRoute

- Other html page file

```
$routeProvider.when(
        url,
        {
            template: string,
            templateUrl: string,
            controller: string,function or array
            controllerAS: string,
            resolve:object<key,function>
        });
```

ngRoute

- 指定要使用的controller
- 也可以直接撰寫controller內容

```
$routeProvider.when(
        url,
        {
            template: string,
            templateUrl: string,
            controller: string,function or array
            controllerAS: string,
            resolve:object<key,function>
        });
```

ngRoute

- 指定controllerAs別名

```
$routeProvider.when(
        url,
        {
            template: string,
            templateUrl: string,
            controller: string,function or array
            controllerAS: string,
            resolve:object<key,function>
        });
```

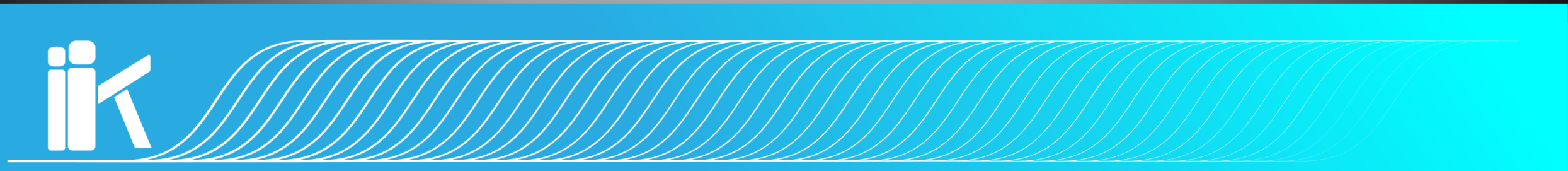
ngRoute

- 在controller & route被載入前執行特定的非同步工作
- 不建議做太費時的任務

```
$routeProvider.when(
        url,
        {
            template: string,
            templateUrl: string,
            controller: string,function or array
            controllerAS: string,
            resolve:object<key,function>
        });
```

Lab

- ngRoute + $http or $resource 實作CRUD
- Web API URL : http://myngwebapi.azurewebsites.net/api/employee



ui-Router

- 透過改變$state進行URL的轉變
- 支援多個view套版
  -
  -
- 支援巢狀view
  - < div ui-view ="view1"></ div >
  - view1.html can use < div ui-view ="view2"></ div >
- 支援透過自訂名稱進行導航

ui-Router

- <div ui-view>

- ui-sref='state-name'
  - <a ui-sref="list">Def Page
  - 依路由設定render成標準的html

ui-Router

- 參數傳遞
  - ui-sref='state-name ({param1: value, param2: value})'
  - <a ui-sref="detail({ id: contact.id })">....
- 直接指定導航
  - $state.go('list');
- 參數取得
  - $staeParams

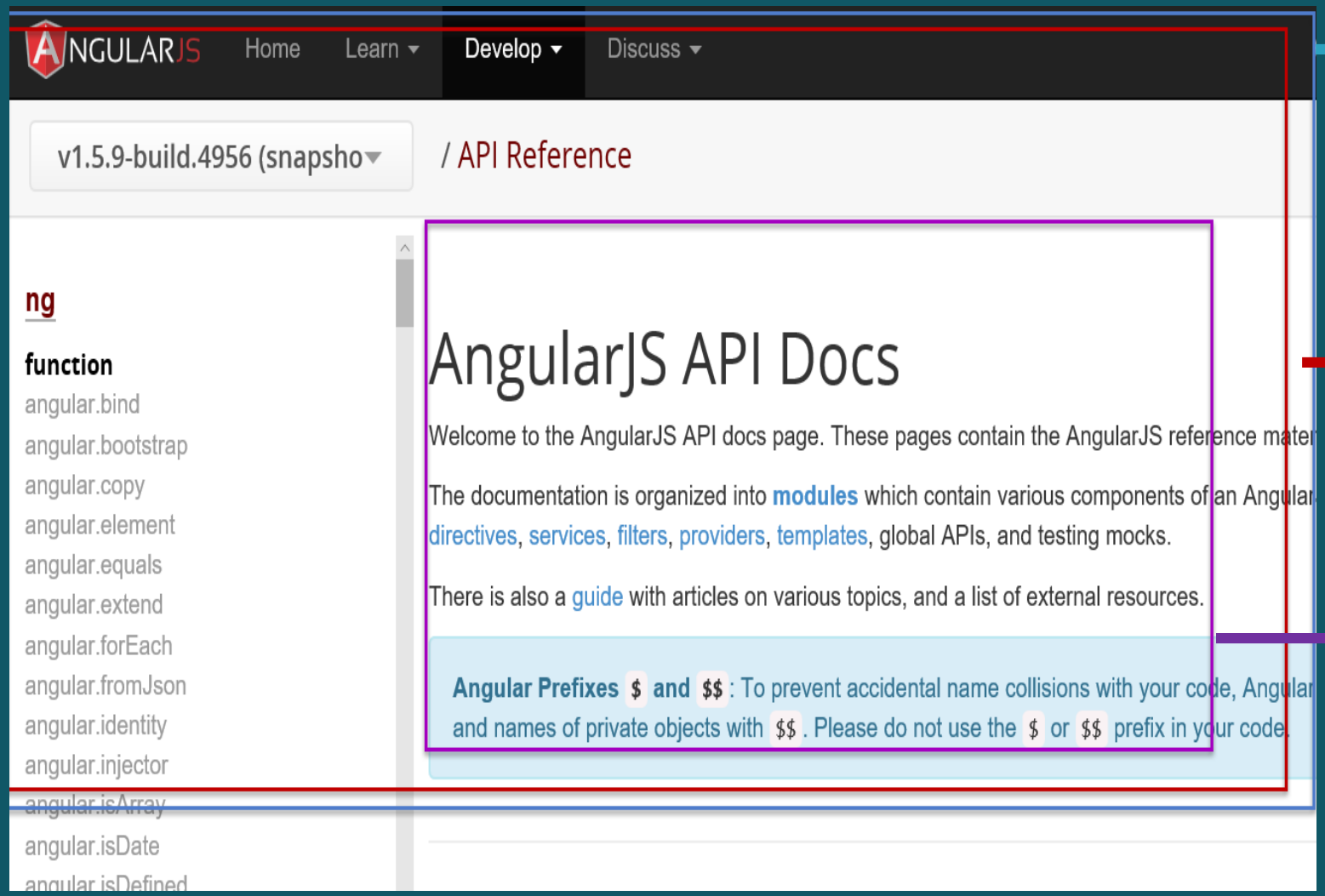
ui-Router

```
var app = angular.module('myapp', ['ui.router']);
app.config(['$stateProvider', '$urlRouterProvider',
        function ($stateProvider, $urlRouterProvider) {

            $urlRouterProvider.otherwise('/');

            $stateProvider.state('list', {
                url: '/',
                templateUrl: 'uirt21.html',
                controller: 'listctrl as ctrl'
            }).state('view', {
                url: '/view/:no',
                templateUrl: 'uirt22.html',
                controller: 'detailctrl as empctrl'
            });
        }]);
```

ui-Router -嵌套路由



配置$stateProvider
ui-view -> index.html

配置左頁功能選單
ui-view -> 功能xx.html

配置左頁功能選單
ui-view -> 功能xx.html

Directives

- ng-model、ng-show 、 ng-disabled 、 ng-bind……
- 建立可重用的HTML – ng-include
  - < div ng-include = " ' views/subview.html ' " >< / div >
  - < div ng-include = " ctrl.subview " >< / div >

自訂Directives

```
<emp-ui></emp-ui>
```

```
<div ng-include="ctrl.empui">
```

```
<div emp-ui></div>
```

```
<div class="emp-ui"></div>
```

自訂Directives

```
app.directive("empUi", [function () {
 return {
        //.....
    };
}]);
```

- emp-ui -> empUi

自訂Directives

- restrict 定義directive的使用方式
  - A -> <div emp-ui> 預設值
  - E -> <emp-ui></emp-ui>
  - C ->
  - 可混用
- templateUrl 指定html template

自訂Directives

- link 函式做為 directive 的核心作業邏輯

```
app.directive("empUi", [function () {
    return {
        restrict: 'AE',
        templateUrl: 'ngdirective11.html',
        link: function (scope, element, attrs) {
            //....
        }
    };
}]);
```

Component

- 1.5版 easier to upgrade to AngularJS 2
- In Angular, a Component is a special kind of directive
- 簡化directive配置，易於使用
- 無法操作DOM (directive link unavailable)
- 採用自定義的HTML Tag
  - <my-component></my-component>

Summary

- AngularJS 一個前端MVC Framework
- 實現SPA頁面應用的利器
- 處理Data Model而非針對DOM
- 不同模組間的服務採用注入方式
- Services / directive / component 實現reuse程式碼

Ref

- https://docs.angularjs.org/tutorial
- http://www.w3schools.com/angular/