Javascript Elements pour Angular JS

Closure

```
(function() {

})();
```

Module

Définir un module

```
angular.module("MyModule",[]);
```

Définir un module et le stocker dans une variable

```
var myModule=angular.module("MyModule",[]);
```

Définir un module avec une dépendance

```
angular.module("MyModule",["anotherModule"]);
```

Faire référence à un module existant

```
var myModule=angular.module("MyModule");
```

Faire référence à un module dans une vue html

```
<div ng-app="MyModule">
...
</div>
```

Controller

Créer un contrôleur dans MyModule

```
angular.module("MyModule").controller("MyController",function(){
...
});
```

Créer un contrôleur dans MyModule utilisant le service $http

```
angular.module("MyModule").controller("MyController",["$http",function($http){
...
}]);
```

Faire référence à un contrôleur dans une vue html

```
<div ng-controller="MyController">
...
</div>
```

Directive

Créer une directive de type Element dans MyModule

```
angular.module("MyModule").directive("MyDirective",function(){
    return {
        restrict:'E',
        template:'<html>....',
        templateUrl:'urlForTemplate',
        controller:function(){
        //Controller code
        },
        controllerAs:'alliasForController'
    };
});
```

Directive complète :

```
myApp.directive( 'myDirective', function () {
    return {
        restrict: 'EA',
        controller: function( $scope, $element, $attrs, $transclude ) {
            // Controller code goes here.
        },
        compile: function compile( tElement, tAttributes, transcludeFn ) {
            // Compile code goes here.
            return {
                pre: function preLink( scope, element, attributes, controller,
transcludeFn ) {
                    // Pre-link code goes here
                },
                post: function postLink( scope, element, attributes, controller,
transcludeFn ) {
                    // Post-link code goes here
                }
            };
        }
    };
});
```

Injection de dépendance (dependency injection)

Elle permet d'utiliser des services ou des modules existants à l'intérieur d'un autre (controller, service...)

Injection avec tableau (conseillé)

```
angular.module("MyModule").controller("MyController", ["$scope", "greeter",
function($scope, greeter) {
  // ...
}]);
```

Injection avec la propriété **$inject**

```
var MyController = function($scope, greeter) {
  // ...
}
MyController.$inject = ["$scope", "greeter"];
angular.module("MyModule").controller("MyController", MyController);
```

Quelque soit la méthode utilisée, veillez à faire en sorte que les paramètres de la fonction controller soient définis dans le même ordre que les services injectés

<fc #FF0000>A éviter : l'injection implicite</fc>

```
angular.module("MyModule").controller("MyController", function($scope, greeter) {
  // ...
});
```

Vue (HTML File)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Angular JS view</title>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.8/angular.min.js"></script>
</head>
<body data-ng-app>
</body>
</html>
```

Création d'objets injectables

Factory

```
monApp.factory('maFactory', function() {
    return {
        "member": "value",
        "method": function(){
            // faire qqchose
        }
    }
});
```

maFactory est un singleton (une seule instance), et devient injectable :

Injection dans un contrôleur :

```
app.controller("monController",["$scope","maFactory",function($scope,maFactory){
    //Utilisation de maFactory
    $scope.member=maFactory.member;
    maFactory.method();
    ...
}]);
```

Service

Le service permet également de créer un objet injectable, mais d'une autre façon :

```
monApp.service('monService', function() {
    this.name = "Anonymous";
    this.id = null;
    this.login = function(){
        // faire un truc pour se connecter
    }
    this.logout = function(){
        // faire un truc pour se déconnecter
    }
});
```