AngularJS--第七天 课程

$anchorScroll：依赖注入这个服务，实现锚点跳转功能。

案例 锚点跳转

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <style>
        #parent div{
            width: 300px;
            height: 600px;
            border:2px solid red;
            margin:20px;
        }
        #parent ul{
            width: 200px;
            position: fixed;
            top: 0;
            right: 0;
        }
    </style>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',['$scope','$location','$anchorScroll',function($scope,$location,$anchorScroll){
        $scope.change = function(id){

        //console.log(id);
        $location.hash(id);
        // 设置哈希值
        $anchorScroll();
        // 不写这个会出现小bug
    };


    }]);

</script>
</head>

<body>
<div id="parent" ng-controller="Aaa">
    <ul>
        <li ng-repeat="id in [1,2,3,4,5]" ng-click="change('div'+id)">{{id}}要跳转的第{{id}}个div</li>
    </ul>
    <div ng-repeat="id in [1,2,3,4,5]" ng-attr-id="div{{id}}">{{id}}</div>
</div>
</body>
</html>
```

$cacheFactory：缓存

info()：查看缓存数据

```
console.log(cache.info());
    // 用info()方法查看缓存数据
```

put()：缓存数据

```
cache.put('name','boy');
      cache.put('age','12');
      // 用put()方法缓存数据
```

get()：通过key值获取value值

```
console.log(cache.get('name'));
      // 用get()方法可以通过key值找到value值
```

remove()：删除缓存

```
 console.log(cache.remove('name'));
      // 删除缓存数据
```

配置capacity：限制存储数量（长度）

```
 var cache = $cacheFactory('myCache',{ capacity: 2 });
      // capacity：限制缓存数量（长度）
```

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <script src="js/angular.min.js"></script>
   <script>
   var m1 = angular.module('myApp',[]);
   m1.controller('Aaa',['$scope','$cacheFactory',function($scope,$cacheFactory){
      // var cache = $cacheFactory('myCache');
      var cache = $cacheFactory('myCache',{ capacity: 2 });
      // capacity：限制缓存数量（长度）
      cache.put('name','boy');
      cache.put('age','12');
      cache.put('ageq','42');
      // 用put()方法缓存数据
      console.log(cache.info());
      // 用info()方法查看缓存数据

      // console.log(cache.get('name'));
      // 用get()方法可以通过key值找到value值
        // console.log(cache.remove('name'));
      // 删除缓存数据

   }]);

</script>
</head>

<body>
<div id="parent" ng-controller="Aaa">

</div>
</body>
</html>
```

$log

log()　打印

```
$log.log('hello');
      // 类似于console.log();打印数据
```

info()　提示

```
$log.info('hello');
```

warn() 警告

```
$log.warn('hello');
```

error() 错误

```
$log.error('hello');
```

$interpolate：插值计算

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <style>
        #parent div{
            width: 300px;
            height: 600px;
            border:2px solid red;
            margin:20px;
        }
        #parent ul{
            width: 200px;
            position: fixed;
            top: 0;
            right: 0;
        }
    </style>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',['$scope','$interpolate',function($scope,$interpolate){
            $scope.$watch("Text2",function(newText){
                // 用$watch监听Text2的数据变化
                    if(newText){
                        // 判断是否有新值
                        var temp = $interpolate(newText);
                        // 当新值存在时，把新值传入$interpolate方法中、放入一个变量
                        $scope.Text3=temp({Text1 : $scope.Text1})
                        // 给p标签中的Text3进行数据绑定、赋值的是文本框中的Text1的内容
                    }
            });
    }]);

</script>
</head>

<body>
<div id="parent" ng-controller="Aaa">
    <input type="text" ng-model="Text1">
    <textarea ng-model="Text2"></textarea>
    <p>{{ Text3 }}</p>
</div>
</body>
</html>
```

$q：对异步操作进行一些功能扩展

promise的实现

$q.defer()：创建一个延迟对象

resolve()：成功时候触发

reject()：失败时候触发

notify()：实时通知

then()：监听时的回调

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',['$scope','$q',function($scope,$q){
            var df = $q.defer();

            function show(){
                setTimeout(function(){
                    df.resolve();
                    // 成功的
                    // df.reject();
                    // 失败的
                },2000);
                return df.promise;
                // 返回一个promise对象
            };
            // 执行show调用then方法
            show().then(function(){
                alert('成功');
                // 第一个参数成功触发
            },function(){
                alert('失败');
                // 第二个参数失败触发
            });
    }]);

</script>
</head>

<body>
<div ng-controller="Aaa">

</div>
</body>
</html>
```

angularJs的供应商：服务的相关初始配置操作

config：实现一些初始化配置

provider：供应商的参数

$interpolate：插入计算

startSymbol()：可以设置前面两个{{的样式

endSymbol()：可以设置后面两个}}的样式

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
```

```
m1.config(["$interpolateProvider",function($interpolateProvider){
    // 在m1的模块下，config方法、接收一个数组引用相应的供应商
    // 首先是服务的名称$interpolate后面加上Provider这样就变成了这个服务的供应商
    // 如果是$scope。那就是$scopeProvider
    // 并不是所有服务都有供应商Provider。大部分有！
    $interpolateProvider.startSymbol('@@');
        // 设置表达式的样式，startSymbol可以设置前面两个{{的样式
    $interpolateProvider.endSymbol('@@');
    // 设置表达式的样式，endSymbol可以设置后面两个}}的样式
}]);
m1.controller('Aaa',['$scope','$interpolate',function($scope,$interpolate){
     $scope.name="hello";
}]);

</script>
</head>

<body>
<div ng-controller="Aaa">
   @@ name @@
</div>
</body>
</html
```

$anchorScroll：依赖注入这个服务，实现锚点跳转功能。

disableAutoScrolling()：禁止自动跳转

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <style>
        #parent div{
            width: 300px;
            height: 600px;
            border:2px solid red;
            margin:20px;
        }
        #parent ul{
            width: 200px;
            position: fixed;
            top: 0;
            right: 0;
        }
    </style>
    <script>
    var m1 = angular.module('myApp',[]);
     m1.config(["$anchorScrollProvider",function($anchorScrollProvider){
        $anchorScrollProvider.disableAutoScrolling();
        // 用disableAutoScrolling()可以禁止哈希值自动跳转
        // 需要调用 $anchorScroll();才能跳转
    }]);
    m1.controller('Aaa',['$scope','$location','$anchorScroll',function($scope,$location,$anchorScroll){
        $scope.change = function(id){

        //console.log(id);
        $location.hash(id);
        // 设置哈希值
        // $anchorScroll();
        // 不写这个会出现小bug
    };
```

```
}]);

</script>
</head>

<body>
<div id="parent" ng-controller="Aaa">
    <ul>
        <li ng-repeat="id in [1,2,3,4,5]" ng-click="change('div'+id)">{{id}}要跳转的第{{id}}个div</li>
    </ul>
    <div ng-repeat="id in [1,2,3,4,5]" ng-attr-id="div{{id}}">{{id}}</div>
</div>
</body>
</html>
```

factory()：自定义服务

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.factory("myServics",function(){
            return {
                name:"hello",
                run : function(){
                    return this.name +':angular';
                }
            }
    });
    // m1.factory()：自定义服务、接收两个参数，第一个是名字，第二个是回调函数
    // 自定义的名字不要以$开头。$开头的是angular内部的服务

    m1.controller('Aaa',['$scope','myServics',function($scope,myServics){

        console.log(myServics.run());
    }]);

</script>
</head>

<body>
<div id="parent" ng-controller="Aaa">

</div>
</body>
</html>
```

provider()：自定义服务

provider()：和factory()的区别是可配置，在配置供应商中可操作的

factory()其实也会调用provider()的。是一种简写方式

$get

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
```

```
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.provider("myServics",function(){
        return{
            name:'hi',
            $get:function(){
                // 嵌套了一层return一层$get
                return {
                        name:this.name,
                        run : function(){
                            return this.name +':angular';
                        }
                }
            }
        }

    });
    m1.config(['myServicsProvider',function(myServicsProvider){
        console.log(myServicsProvider);
        myServicsProvider.name="fengniao";
    }])
    m1.controller('Aaa',['$scope','myServics',function($scope,myServics){

            console.log(myServics.run());

    }]);

</script>
</head>

<body>
<div id="parent" ng-controller="Aaa">

</div>
</body>
</html>
```

随机数四舍五入

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.provider('myMath',function(){
        return{
            btInt:false,
            // 设置一个开关
            int:function(oInt){
                // 判断传入的是true还是false
                if(oInt){
                    this.btInt=true;
                }else{
                    this.btInt=false;
                }
            },
            $get:function(){
                var This=this;
                // 存储this指向
                return function(num1,num2){
```

```
                return This.btInt ? Math.round(Math.random()*(num2-num1)+num1):Math.random()*(num2-num1)+num1;
                // 返回一个三目的结果~
            }
          }
        }
    })
  m1.config(['myMathProvider',function(myMathProvider){
      myMathProvider.int(false);
      // 在myMath自定义服务的供应商中调用int方法！并传入参数true或者false
      // 当为true是时四舍五入、为false时候是正常的随机数
  }])
    m1.controller('Aaa',["$scope","myMath",function($scope,myMath){
      console.log(myMath(-5,9));
  // 打印自定义服务myMath(-5,9)并传两个参数

    }]);
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
        <p>{{ name }}</p>
    </div>
</body>
</html>
```