# angularjs入门

## AngularJS是什么?

### Google开源的前端JS结构化框架

·https://angularjs.org/

### AngularJS特性和优点

·双向数据绑定

·声明式依赖注入

·解耦应用逻辑, 数据模型和视图

·完善的页面指令

·定制表单验证

·Ajax封装

### 与jQuery的比较

·jQuery

- JS函数库

- 封装简化dom操作

·angular

- JS结构化框架

- 主体不再是DOM, 而是页面中的动态数据

### AngularJS能做什么项目

·构建单页面(SPA)Web应用或Web App应用

- 将所有的活动局限于一个页面

- 当页面中有部分数据发生了变化，不会去刷新整个页面，而是局部刷新

- 利用的是ajax技术和路由

·微信网页版

- <https://wx.qq.com/>

·后台管理应用

- 阿里云, 土豆后台, 唯品会...

## 开发第一个Angular应用

·当文本框输入时，在页面实时显示输入的内容

### jQuery实现

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title></title>  
</head>  
<body>  
<input type="text">  
<p>您输入的内容是:<span></span></p>  
<script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.6.0/jquery.min.js"></script>  
<script type="text/javascript">  
 $(function () {//document.ready文档(结构)加载完毕 window.onload:文档和数据加载完毕  
 //当元素的值发生改变时，会发生 change 事件,失去焦点时才会调用。  
 //keyup键盘按下响应事件  
 $("input").keyup(function () {  
 // var value = this.value;  
 var value = $(this).val();  
 $("span").html(value);  
 })  
 })  
</script>  
</body>  
</html>

### angular

·ng-app

- 告诉angular核心它管理当前标签所包含的整个区域,并且会自动创建$rootScope根作用域对象

·ng-model

- 将当前输入框的值与谁关联(属性名:属性值), 并作为当前作用域对象($rootScope)的属性

·{{}} (表达式)

- 显示数据,从作用域对象的指定属性名上取

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title></title>  
</head>  
<!--ng-app(指令) : 告诉angular核心它管理当前标签所包含的整个区域,并且会自动创建$rootScope根作用域对象-->  
<body ng-app>  
<!--  
 ng-model : 将当前输入框的值与谁关联(属性名:属性值), 并作为当前作用域对象($rootScope)的属性  
 {{}} (表达式) : 显示数据,从作用域对象的指定属性名上取  
-->  
<!--将输入框的ng-model值设为username-->  
<input type="text" ng-model="username">  
<!--span标签通过{{username}}绑定输入框-->  
<p>您输入的内容是:<span>{{username}}</span></p>  
<!--引入本地angular.js文件-->  
<!--<script src="./js/angular-1.2.29/angular.js"></script>-->  
<!--引入跨域angular.js文件-->  
<script src="https://cdn.bootcdn.net/ajax/libs/angular.js/1.8.2/angular.min.js"></script>  
</body>  
</html>

### 表达式和语句的区别

·表达式

- 通常有一个返回值，可以放在任何需要值得地方，比如函数调用的参数，一个变量名，一个运算

·语句

- 通常表示一个完整的执行单位，一段完整的js可执行的代码，有的语句也可以用表达式来执行，叫做表达式语句

·区别

- 语句用分号结尾，有些语句我们没有加分号，比如console.log虽然我们没有加分号，但也是语句，因为js引擎会自动解析并且加上分号。

·特例

- if语句，就不用加封号，可也是完整的语句

# 双向数据绑定

## 数据绑定

·数据从一个地方A转移(传递)到另一个地方B, 而且这个操作由框架来完成

·视图(View)

- 也就是我们的页面(主要是Andular指令和表达式)

·模型(Model)

- 域对象(当前为$rootScope), 它可以包含一些属性或方法

·当改变View中的数据, Model对象的对应属性也会随之改变

·当Model域对象的属性发生改变时, 页面对应数据随之更新

## 双向数据绑定

·数据可以从View流向Model, 也可以从Model流向View

·ng-model是双向数据绑定, 而{{}}是单向数据绑定

·ng-init

- 用来初始化当前作用域变量

- 单向的，从view流向model

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>双向数据绑定</title>  
</head>  
<body ng-app ng-init="username = '初始化数据'">  
<input type="text" ng-model="username">  
<p>姓名1：{{username}}</p>  
<input type="text" ng-model="username">  
<p>姓名2：{{username}}</p>  
  
<script src="./js/angular-1.2.29/angular.js"></script>  
</body>  
</html>

# 作用域与控制器

## 作用域(scope)

·一个js实例对象, ng-app指令默认会创建一个根作用域对象($rootScope)

·它的属性和方法与页面中的指令或表达式是关联的

## 控制器(controller)

·用来控制AngularJS应用数据的实例对象

·ng-controller

- 指定控制器构造函数, Angular会自动new此函数创建控制器对象

·同时Angular还有创建一个新的域对象$scope, 它是$rootScope的子对象

·在控制器函数中声明$scope形参, Angular会自动将$scope传入

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>作用域与控制器</title>  
</head>  
<body ng-app>  
<!--指定控制器构造函数, Angular会自动new此函数创建控制器对象-->  
<div ng-controller="MyController">  
<input type="text" ng-model="firstName">  
<input type="text" ng-model="lastName">  
<p>姓名1：{{firstName+"-"+lastName}}</p>  
 <!--调用$scope里面的getName方法-->  
<p>姓名2：{{getName()}}</p>  
</div>  
<script src="./js/angular-1.2.29/angular.js"></script>  
<script>  
 //在控制器函数中声明$scope形参, Angular会自动将$scope传入  
 function MyController($scope) {//形参必须是$scope  
 // console.log(this instanceof MyController);//true  
 // console.log($scope);  
 //初始化firstName和lastName  
 $scope.firstName = "kobe";  
 $scope.lastName = "bryant";  
 //添加gerName方法  
 $scope.getName = function () {  
 //将值返回  
 // return $scope.firstName+" "+$scope.lastName;  
 return this.firstName+" "+this.lastName;  
 }  
 }  
</script>  
</body>  
</html>

# 依赖注入

·依赖对象

- 完成某个特定的功能需要某个对象才能实现，这个对象就是依赖对象

·依赖注入

- 依赖的对象以形参的形式被注入进来使用，这种方式就是声明式依赖注入

·angular的 ‘$scope’对象就是依赖对象，并且是依赖注入的形式进行使用

·形参必须是特定的名称, 否则Angular无法注入抛异常

·回调函数的event的就是依赖对象

·回调函数有形参就是依赖注入

## 开发的两种方式

·需求：将数组里的每一项值加10输出

·命令式

- 更加注重执行的过程

//命令式  
var ***arr*** = [1,2,3,4,5];  
var ***newArr*** = [];  
for(var ***i*** = 0;***i***<***arr***.length;***i***++){  
 var ***value*** = ***arr***[***i***] + 10;  
 ***newArr***.push(***value***);  
}  
***console***.log(***newArr***);//[11, 12, 13, 14, 15]

·声明式

- 更加注重执行的结果

- 声明式是对命令式的局部包装

//声明式,map可以对数组的每一个值加工并返回  
var ***arr*** = [1,2,3,4,5];  
let ***newArr2*** = ***arr***.map(function (item,index) {  
 return item+10;  
})  
***console***.log(***newArr2***);//[11, 12, 13, 14, 15]

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title></title>  
</head>  
  
<body>  
  
<button id="btn">点我一下</button>  
<script type="***text***/javascript" src="./js/angular-1.2.29/angular.js"></script>  
<script type="***text***/javascript">  
 ***window***.onload = function () {  
 //这里传入的event就是依赖对象，这种方式叫依赖注入  
 ***document***.getElementById('btn').onclick = function (event) {  
 alert(event.clientX);  
 }  
 }  
</script>  
  
<script type="***text***/javascript">  
 //命令式  
 var ***arr*** = [1,2,3,4,5];  
 var ***newArr*** = [];  
 for(var ***i*** = 0;***i***<***arr***.length;***i***++){  
 var ***value*** = ***arr***[***i***] + 10;  
 ***newArr***.push(***value***);  
 }  
 ***console***.log(***newArr***);//[11, 12, 13, 14, 15]  
 //声明式,map可以对数组的每一个值加工并返回  
 let ***newArr2*** = ***arr***.map(function (item,index) {  
 return item+10;  
 })  
 ***console***.log(***newArr2***);//[11, 12, 13, 14, 15]  
</script>  
</body>  
</html>

# 模块对象

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title></title>  
</head>  
<!--引入模块对象-->  
<body ng-app="myApp">  
<!--引入作用域对象-->  
<div ng-controller="myController">  
 <input type="text" ng-model="empName">  
 <p>员工名字1:{{empName}}</p>  
<!-- <input type="text" ng-model="empName1">-->  
<!-- <p>员工名字2:{{empName1}}</p>-->  
</div>  
<div ng-controller="myController1">  
 <input type="text" ng-model="empName">  
 <p>员工名字2:{{empName}}</p>  
</div>  
<script type="***text***/javascript" src="./js/angular-1.5.5/angular.js"></script>  
<script type="***text***/javascript">  
 //创建模块对象  
 //第一个参数是模块对象的名字  
 //第二参数是依赖，没有则写空数组  
 // var myModule = angular.module("myApp",[]);  
 // //生成作用域对象  
 // //传入作用域对象名和回调函数  
 // myModule.controller("myController",function ($scope) {  
 // $scope.empName = "kobe";  
 // // $scope.empName1 = "张三";  
 // });  
 // myModule.controller("myController1",function ($scope) {  
 // $scope.empName = "张三";  
 // });  
  
 //优化，链式调用  
 // angular.module("myApp",[])  
 // .controller("myController",function ($scope) {//返回值是模块对象  
 // $scope.empName = "kobe";  
 // // $scope.empName1 = "张三";  
 // })  
 // .controller("myController1",function ($scope) {//隐式声明依赖注入  
 // $scope.empName = "张三";  
 // });  
   
 //以上的代码有问题：  
 //js代码压缩后形参会被abcd字母代替  
 //代码压缩的$scope会被abcd代替，angular解析不了  
 //解决方案  
 //传入一个数组，里面有形参值和回调函数  
 //会自动使用数组的形参值代替回调函数的形参值  
 //这种解决方案叫显式声明依赖注入  
 angular.*module*("myApp",[])  
 .controller("myController",["$scope",function ($scope) {  
 $scope.empName = "kobe";  
 }])  
 .controller("myController1",["$scope",function ($scope) {  
 $scope.empName = "张三";  
 }]);  
</script>  
</body>  
</html>

# 表达式

·使用Angular表达式

- 语法

-- {{expression}}

- 作用

-- 显示表达式的结果数据

- 注意

-- 表达式中引用的变量必须是当前域对象有的属性(包括其原型属性)

·操作的数据

- 基本类型数据

-- number/string/boolean

- undefined, Infinity, NaN, null解析为空串: "", 不显示任何效果

- 对象的属性或方法

- 数组

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title></title>  
</head>  
<body ng-app="myApp">  
<div ng-controller="myCtr1">  
 <!--操作number-->  
 <p>{{123}}</p>  
 <!--操作字符串-->  
 <p>{{"abc"}}</p>  
 <!--操作变量-->  
 <p>{{abc}}</p>  
 <!--boolean值-->  
 <p>{{false}}</p>  
 <!--undefined, Infinity, NaN, null解析为空串: "", 不显示任何效果-->  
 <p>{{undefined}}</p>  
 <!--对象的属性或方法-->  
 <p>{{obj.name}}</p>  
 <p>{{obj.say()}}</p>  
 <!--数组-->  
 <p>{{***arr***}}</p>  
</div>  
  
</body>  
<script type='***text***/javascript' src="./js/angular-1.5.5/angular.js"></script>  
<script>  
 //创建模块对象  
 angular.*module*("myApp",[])  
 .controller("myCtr1",["$scope",function ($scope) {//参数可以不为$scope  
 $scope.abc = "我是abc";  
 $scope.obj = {  
 name:"孙悟空",  
 say:function () {  
 return "我是孙大圣";  
 }  
 };  
 $scope.arr = [1,2,3,4,5];  
 }])  
</script>  
</html>

# 常用指令

·Angular为HTML页面扩展的自定义标签属性或标签

·与Angular的作用域对象(scope)交互,扩展页面的动态表现力

·ng-app

- 指定模块名，angular管理的区域

<body ng-app="myApp" ></body>

·ng-model

- 双向绑定，输入相关标签

数量：<input type="number" ng-model="count1">

·ng-init

- 初始化数据

<body ng-app="myApp" ng-init="sex='男’"></body>

·ng-click

- 调用作用域对象的方法（点击时）, 可以传$event

<button ng-click="getTotalPrice()">计算</button>

$scope.getTotalPrice = function () {  
 $scope.totalPrice = $scope.price2\*$scope.count2;  
 ***console***.log($scope.totalPrice);  
};

·ng-controller

- 指定控制器构造函数名，内部会自动创建一个新的子作用域（外部的）

<div ng-controller="myCtr1">

·ng-bind

- 解决使用{{}}显示数据闪屏（在很短时间内显示{{}}）

<div>  
 <!--ng-bind： 解决使用{{}}显示数据闪屏（在很短时间内显示{{}}）-->  
 <p ng-bind="shanpin"></p>  
</div>

//解决闪屏问题  
$scope.shanpin = "解决闪屏问题";

·ng-repeat

- 遍历数组显示数据， 数组有几个元素就会产生几个新的作用域

- $index

-- 数组下标

- $first

-- 判断是否为第一个元素

- $last

-- 判断是否为最后一个元素

- $middle

-- 除了第一个和最后一个元素为false，其他为true

- $odd

-- 下标奇数为true

- $even

-- 下标偶数为true

·ng-class

- 动态引用定义的样式 {aClass:true, bClass:false}

<ul>  
 <!--ng-repeat： 遍历数组显示数据， 数组有几个元素就会产生几个新的作用域，persons为数组-->  
 <!--ng-class: 动态引用定义的样式 {aClass:true, bClass:false}-->  
 <li ng-repeat="person in persons" ng-class="{oddB:$odd,evenB:$even}">  
 <!--显示获取到的值-->  
 {{$index}}--姓名：{{person.username}}，年龄：{{person.age}}  
 </li>  
</ul>

//初始化人员信息列表  
$scope.persons = [  
 {username:"kobe",age:39},  
 {username:"kobe2",age:38},  
 {username:"kobe3",age:37},  
 {username:"kobe4",age:36},  
 {username:"kobe5",age:35}  
];

<style>  
 .evenB {  
 background-color: yellow;  
 }  
  
 .oddB {  
 background-color: green;  
 }  
 </style>

·ng-show

- 布尔类型， 如果为true才显示

<!--ng-show: 布尔类型， 如果为true才显示-->  
<div>  
 <p ng-show="istrue(true)">show</p>  
 <p ng-show="istrue(false)">noshow</p>  
</div>

//show/hide  
$scope.istrue = function (istrue) {  
 return istrue;  
};

·ng-hide

- 布尔类型， 如果为true就隐藏

<!--ng-hide: 布尔类型， 如果为true才隐藏-->  
<div>  
 <p ng-hide="istrue(true)">hide</p>  
 <p ng-hide="istrue(false)">nohide</p>  
</div>

//show/hide  
$scope.istrue = function (istrue) {  
 return istrue;  
};

·ng-style

- 动态引用通过js指定的样式对象 {color:'red', background:'blue'}

·ng-mouseenter

- 鼠标移入监听, 值为函数调用, 可以传$event

·ng-mouseleave

- 鼠标移出监听, 值为函数调用, 可以传$event

<!--ng-style: 动态引用通过js指定的样式对象 {color:'red', background:'blue'}-->  
<!--ng-mouseenter: 鼠标移入监听, 值为函数调用, 可以传$event-->  
<!--ng-mouseleave: 鼠标移出监听, 值为函数调用, 可以传$event-->  
<div ng-style="myStyle" ng-mouseenter="enter()" ng-mouseleave="leave()">

//ng-style  
$scope.myStyle = {  
 width:"200px",  
 height:"200px",  
 backgroundColor:"red"  
}  
  
//鼠标移入事件  
$scope.enter = function () {  
 //修改myStyle的样式  
 this.myStyle = {  
 width:"180px",  
 height:"180px",  
 backgroundColor: "pink"  
 }  
}  
  
//鼠标移出事件  
$scope.leave = function () {  
 //修改myStyle的样式  
 this.myStyle = {  
 width:"200px",  
 height:"200px",  
 backgroundColor:"red"  
 }  
}

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title></title>  
 <style>  
 ul{  
 list-style-type: none;  
 padding-inline-start: 10px;  
 }  
 .evenB {  
 background-color: yellow;  
 }  
  
 .oddB {  
 background-color: green;  
 }  
 </style>  
</head>  
  
<body ng-app="myApp" ng-init="sex='男'">  
<div ng-controller="myCtr1">  
 <div>  
 <h2>总价计算器（自动）</h2>  
 数量：<input type="number" ng-model="count1">  
 单价：<input type="number" ng-model="price1">  
 <p>总价：{{count1\*price1}}</p>  
 </div>  
  
 <div>  
 <h2>总价计算器（手动）</h2>  
 数量：<input type="number" ng-model="count2">  
 单价：<input type="number" ng-model="price2">  
 <!--此方法不可用-->  
<!-- <button id="btn1">计算</button>-->  
<!-- <p>总价：{{totalPrice}}</p>-->  
 <!--ng-click调用作用域对象的方法（点击时）-->  
 <button ng-click="getTotalPrice()">计算</button>  
 <p>总价：{{totalPrice}}</p>  
 </div>  
  
 <div>  
 <h2>人员信息列表</h2>  
 <ul>  
 <!--ng-repeat： 遍历数组显示数据， 数组有几个元素就会产生几个新的作用域，persons为数组-->  
 <!--ng-class: 动态引用定义的样式 {aClass:true, bClass:false}-->  
 <li ng-repeat="person in persons" ng-class="{oddB:$odd,evenB:$even}">  
 <!--显示获取到的值-->  
 {{$index}}--姓名：{{person.username}}，年龄：{{person.age}}  
 </li>  
 </ul>  
 </div>  
  
 <div>  
 <!--ng-bind： 解决使用{{}}显示数据闪屏（在很短时间内显示{{}}）-->  
 <p ng-bind="shanpin"></p>  
 </div>  
  
 <!--ng-show: 布尔类型， 如果为true才显示-->  
 <div>  
 <p ng-show="istrue(true)">show</p>  
 <p ng-show="istrue(false)">noshow</p>  
 </div>  
  
 <!--ng-hide: 布尔类型， 如果为true才隐藏-->  
 <div>  
 <p ng-hide="istrue(true)">hide</p>  
 <p ng-hide="istrue(false)">nohide</p>  
 </div>  
  
 <!--ng-style: 动态引用通过js指定的样式对象 {color:'red', background:'blue'}-->  
 <!--ng-mouseenter: 鼠标移入监听, 值为函数调用, 可以传$event-->  
 <!--ng-mouseleave: 鼠标移出监听, 值为函数调用, 可以传$event-->  
 <div ng-style="myStyle" ng-mouseenter="enter()" ng-mouseleave="leave()">  
  
 </div>  
  
  
</div>  
  
<script type='***text***/javascript' src='./js/angular-1.5.5/angular.js'></script>  
<script type='***text***/javascript'>  
 //价格计算器  
 //创建模块对象  
 angular.*module*("myApp",[])  
 .controller("myCtr1",["$scope",$scope=>{  
 $scope.price1 = 20;  
 $scope.count1 = 1;  
  
 $scope.price2 = 30;  
 $scope.count2 = 1;  
 $scope.totalPrice = $scope.price2\*$scope.count2;  
 //不能用点击事件方法  
 // document.getElementById("btn1").onclick = function () {  
 // $scope.totalPrice = $scope.price2\*$scope.count2;  
 // console.log($scope.totalPrice);  
 // }  
 $scope.getTotalPrice = function () {  
 $scope.totalPrice = $scope.price2\*$scope.count2;  
 ***console***.log($scope.totalPrice);  
 };  
  
 //初始化人员信息列表  
 $scope.persons = [  
 {username:"kobe",age:39},  
 {username:"kobe2",age:38},  
 {username:"kobe3",age:37},  
 {username:"kobe4",age:36},  
 {username:"kobe5",age:35}  
 ];  
  
 //解决闪屏问题  
 $scope.shanpin = "解决闪屏问题";  
  
 //show/hide  
 $scope.istrue = function (istrue) {  
 return istrue;  
 };  
  
 //ng-style  
 $scope.myStyle = {  
 width:"200px",  
 height:"200px",  
 backgroundColor:"red"  
 }  
  
 //鼠标移入事件  
 $scope.enter = function () {  
 //修改myStyle的样式  
 this.myStyle = {  
 width:"180px",  
 height:"180px",  
 backgroundColor: "pink"  
 }  
 }  
  
 //鼠标移出事件  
 $scope.leave = function () {  
 //修改myStyle的样式  
 this.myStyle = {  
 width:"200px",  
 height:"200px",  
 backgroundColor:"red"  
 }  
 }  
 }])  
</script>  
  
</body>  
</html>

# H5本地存储技术

## sessionStorage

·本地存储技术

·会话存储

·生命周期

- 浏览器打开到关闭

·大小

- 5M

·存储位置

- 浏览器

·setItem(“key”,value)

- 存储数据

·getItem(“key”)

- 获取数据

·removeItem(“key”)

- 删除数据

## localStorage

·本地存储技术

·生命周期

- 永久，除非清空浏览器数据

·大小

- 5M甚至更大

·存储位置

- 浏览器

·setItem(“key”,value)

- 存储数据

·getItem(“key”)

- 获取数据

·removeItem(“key”)

- 删除数据

## cookie

·作用

- 用于浏览器和服务器之间的通信

·特点

- 大小

-- 4kb

- 每次发送请求都携带，导致占用带宽

- 保存在浏览器端

- cookie容易被截取，不安全

·生命周期

- 会话cookie

-- 浏览器打开到关闭的过程

- 人为设置cookie

-- 认为设置时间

# 笔记练习

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>笔记</title>  
</head>  
<body ng-app="myApp">  
 <div ng-controller="myCtr1">  
 <h2>我的笔记</h2>  
 <textarea cols="30" rows="10" ng-model="***message***"></textarea>  
 <div>  
 <button ng-click="save()">保存</button>  
 <button ng-click="read()">读取</button>  
 <button ng-click="remove()">删除</button>  
 </div>  
 <p>剩余的字数：{{getCount()}}</p>  
 <p style="color: cornflowerblue;display: none" id="savetxt"></p>  
 </div>  
</body>  
<script src="js/angular-1.5.5/angular.js"></script>  
<script src="js/app.js"></script>  
</html>

const ***savetxt*** = ***document***.getElementById("savetxt");  
  
angular.*module*("myApp",[])  
 .controller("myCtr1",["$scope",function ($scope) {  
 //初始化文本框数据  
 $scope.message = "";  
 //定义计算剩余字数的方法  
 $scope.getCount = function () {  
 //判断用户输入的内容长度  
 if ($scope.message.length > 200){  
 //截取字符串，并赋值给自身  
 $scope.message = $scope.message.slice(0,200);  
 }  
 //减去文本框字符串长度  
 return 200 - $scope.message.length;  
 }  
  
 //保存数据的方法  
 $scope.save = function () {  
 //将数据转化为json  
 let note\_value = ***JSON***.stringify($scope.message);  
 ***localStorage***.setItem("note\_key",note\_value);  
 //清空文本框  
 $scope.message = "";  
 //设置savetxt文本  
 ***savetxt***.innerHTML = "笔记已保存";  
 //显示笔记已保存  
 ***savetxt***.style.display = "block";  
 //设置定时器  
 setTimeout(()=>{  
 //隐藏笔记已保存  
 ***savetxt***.style.display = "none";  
 },1000);  
 };  
  
 //定义读取数据的方法  
 $scope.read = function () {  
 //获取json数据  
 let note\_value = ***localStorage***.getItem("note\_key");  
 //如果note\_value不为空，将json数据转为js数据  
 note\_value ? note\_value = ***JSON***.parse(note\_value) : note\_value = "";  
 //将数据显示到页面  
 $scope.message = note\_value;  
 }  
  
 //定义删除数据的方法  
 $scope.remove = function () {  
 //删除数据  
 ***localStorage***.removeItem("note\_key");  
 //清空文本框  
 $scope.message = "";  
 //设置savetxt文本  
 ***savetxt***.innerHTML = "笔记已删除";  
 //显示笔记已保存  
 ***savetxt***.style.display = "block";  
 //设置定时器  
 setTimeout(()=>{  
 //隐藏笔记已保存  
 ***savetxt***.style.display = "none";  
 },1000);  
 }  
 }]);

# 备忘录练习

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>备忘录</title>  
</head>  
<body ng-app="myApp">  
<div ng-controller="myCtr1">  
 <h2>我的备忘录</h2>  
 <div>  
 <input type="text" ng-model="newTodo">  
 <button ng-click="add()">添加</button>  
 </div>  
 <div ng-repeat="todo in todos">  
 <input type="checkbox" ng-model="todo.isChecked">  
 <span>{{todo.name}}</span>  
 </div>  
 <button ng-click="del()">删除选中的记录</button>  
</div>  
<script src="./js/angular-1.5.5/angular.js"></script>  
<script src="./js/beiwanglu.js"></script>  
</body>  
</html>

angular.*module*("myApp",[])  
 .controller("myCtr1",["$scope",function ($scope) {  
 //初始化文本框  
 $scope.newTodo = "";  
 //初始化多选框数组  
 $scope.todos = [  
 {name:"吃饭",isChecked:false},  
 {name:"睡觉",isChecked:true},  
 {name:"打豆豆",isChecked:false}  
 ];  
  
 //添加  
 $scope.add = function () {  
 if ($scope.newTodo !== ""){//判断文本框是否为空  
 //获取要添加的数据，并封装成对象  
 let obj = {  
 name:$scope.newTodo,  
 isChecked:false  
 }  
 //将数据添加到数组，unshift是添加到数组最前面  
 $scope.todos.unshift(obj);  
 //清空文本框  
 $scope.newTodo = "";  
 }  
 };  
  
 //删除功能实现  
 $scope.del = function () {  
 // //遍历数组，以下方法不可行  
 // $scope.todos.forEach(function (item,index) {  
 // //找到选中的元素  
 // if (item.isChecked){//如果元素被选中  
 // //使用splice删除数组元素，第一个是起始元素下标，第二个是删除个数  
 // $scope.todos.splice(index,1);  
 // }  
 // });  
  
 //新方法  
 //定义一个变量临时保存数组  
 let oldTodos = $scope.todos;  
 //把原数组清空  
 $scope.todos = []  
 //遍历临时数组  
 oldTodos.forEach(function (item,index) {  
 if (!item.isChecked) {//如果元素没被选中  
 //将元素添加给原数组  
 $scope.todos.push(item);  
 }  
 });  
 };  
 }])