附加：

function sidEffecting(ary) {  
 ary[0] = ary[2];  
}  
function bar(a,b,c=3) {  
 c = 10;  
 sidEffecting(arguments);  
 return a + b + c;  
}  
alert(bar(1,1,1));//12  
  
function sidEffecting(ary) {  
 ary[0] = ary[2];  
}  
function bar(a,b,c) {  
 c = 10;  
 sidEffecting(arguments);  
 return a + b + c;  
}  
alert(bar(1,1,1));//21  
  
var a = [0];  
if ([0]) {  
 console.log(a == true);//false  
} else {  
 console.log("wut");  
}  
  
(function(){  
 var x = y = 1;  
})();  
console.log(y);//1  
console.log(x);  
  
function foo(a) {  
 var a;  
 return a;  
}  
function bar(a) {  
 var a = 'bye';  
 return a;  
}  
alert([foo('hello'), bar('hello')]);//hello,bye  
  
var myRe = /ab\*/g;  
var str = 'abbcdefabh';  
var myArray;  
while ((myArray = myRe.exec(str)) !== null) {  
 var msg = 'Found ' + myArray[0] + '. ';  
 msg += 'Next match starts at ' + myRe.lastIndex;  
 console.log(msg);  
}  
//Found abb. Next match starts at 3  
//Found ab. Next match starts at 9  
  
function captureOne(re, str) {  
 var match = re.exec(str);  
 return match && match[1];  
}  
var numRe = /num=(\d+)/ig,  
 wordRe = /word=(\w+)/i,  
 a1 = captureOne(numRe, "num=1"),  
 a2 = captureOne(wordRe, "word=1"),  
 a3 = captureOne(numRe, "NUM=2"),  
 a4 = captureOne(wordRe, "WORD=2");  
alert([a1 === a2, a3 === a4]);//true,false

1、

//字符创、数值、布尔 按值传递引用类型  
//数组、对象、函数 按址传递（他们是引用类型）  
function append(arr,item){  
 var arr1=[];  
 arr1=arr;  
 arr1.push(item);  
 alert(arr);//[1,2,3,4,10]  
 return arr1;  
}  
append([1,2,3,4],10);

2、

<meta name="viewport" content="initial-scale=1,user-scalable=no,width=device-width">

<link href="../css/ionic.min.css" rel="stylesheet">  
<link href="../css/ionicons.min.css" rel="stylesheet">

3、ionic:是一个强大的移动应用程序开发框架  
特点：  
 1）ionic基于angular语法  
 2）ionic是一个轻量级的框架  
 3）性能优越，运算速度快  
 4）ionic专注原生开发--可利用html5\css3\javascript构建接近原生体验的移动应用程序  
①native（原生）  
②webapp  
③hybrid --- 介于原生与webapp之间的技术 --- phonegap(cordova)  
ionic分为三部分：  
 1）css框架  
 2）js框架：基于angular.js  
 3）命令行

4、

<!--ionic css部分-->  
<!--ionic 头部和底部-->  
<!--  
bar：将元素声明为屏幕上绝对定位的块状区域，固定的高度44px  
bar-header：置顶  
bar-footer：置底  
  
副标题：  
bar-subheader  
bar-subfooter  
  
图标：ionicons采用了TrueType字体实现图形样式，超过500种  
icon：将元素声明为图标  
ion-具体图标名称 如：ion-home  
  
按钮：ionic  
-->

<div class="bar bar-header bar-positive">  
 <h1 class="title">...head title...</h1>  
 <!--图标按钮-->  
 <button class="button button-calm icon ion-home icon-right">button01</button>  
</div>  
  
<!--普通按钮-->

//button-block 完全填充父元素宽度  
//button-full 按钮显示完全宽度 不包括内边距  
//button-small  
//button-large  
//button-outline 无背景按钮（背景透明），有边框  
//button-clear 无背景无边框  
<button class="button button-assertive">button01</button>  
<button class="button button-large button-calm">button01</button>  
  
<!--按钮栏-->  
<div class="button-bar">  
 <a class="button button-clear button-calm">button01</a>  
 <a class="button button-calm">button02</a>  
 <a class="button button-calm">button03</a>  
</div>  
  
<!--内容-->  
//has-header 判断是否有主标题  
//has-subheader 判断是否有副标题  
<div class="scroll-content has-header">  
 aaaaaaaaaaaaa  
</div>  
  
<!--图标-->  
<div style="font-size: 40px;color:#008000;"><i class="icon ion-home d1"></i></div>  
  
<div class="bar bar-footer bar-assertive">  
 <div class="title">...footer title...</div>  
</div>

5、

<div class="content scroll-content has-header" style="overflow: scroll;">  
 <!--ionic 列表-->  
 <!--普通列表-->  
 <ul class="list">  
 <li class="item">aaaa</li>  
 <li class="item">bbbb</li>  
 <li class="item">cccc</li>  
 <li class="item">dddd</li>  
 </ul>  
  
 <div class="list">  
 <a class="item">aaaa</a>  
 <a class="item">bbbb</a>  
 <!--列表分割符-->  
 <div class="item item-divider">qqqqqq</div>  
 <!--带图标列表-->  
 <!--左面-->  
 <a class="item item-icon-right">  
 <div class="icon ion-happy"></div>  
 cccc  
 </a>  
 <!--左右都有-->  
 <a class="item item-icon-left item-icon-right">  
 <div class="icon ion-happy"></div>  
 dddd  
 <div class="icon ion-happy"></div>  
 </a>  
 <!--item-note 注释-->  
 <a class="item">  
 aaaa  
 <span class="item-note">消息</span>  
 </a>  
 <!--badge 标记 badge-assertive 标记颜色-->  
 <a class="item">  
 aaaa  
 <span class="badge badge-assertive">12</span>  
 </a>  
 </div>  
  
 <div class="list">  
 <!--带头像的列表-->  
 //item-avatar 创建头像列表，默认设置为40\*40  
   
 <a class="item item-avatar">  
 <img src="images/Dior01.jpg">  
 <h2>aaaa</h2>  
 <p>bbbbbbbb</p>  
 </a>  
 <!--缩略图列表-->  
 <a class="item item-thumbnail-left">  
 <img src="images/Dior02.jpg">aaaaa  
 </a>  
 <a class="item item-thumbnail-right">  
 aaaaa<img src="images/Dior03.jpg">  
 </a>  
 <!--加载大图-->  
 <a class="item">  
 <img src="images/Dior04.jpg">  
 </a>  
 </div>  
 <!--内嵌列表 list-inset-->  
 <div class="list list-inset">  
 <a class="item">aaaa</a>  
 <a class="item">bbbb</a>  
 <a class="item">aaaa</a>  
 <a class="item">bbbb</a>  
 </div>  
  
 <!--ionic 卡片-->  
 //默认自带阴影  
 //item-text-wrap 文本自动折行  
 //item-divider 分割  
   
 <div class="card">  
 <!--分割-->  
 <div class="item item-divider">header card</div>  
 <div class="item item-text-wrap">

ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd</div>  
 <div class="item">aaaa</div>  
 <div class="item">aaaa</div>  
 <div class="item">aaaa</div>  
 <div class="item item-divider">footer card</div>  
 </div>  
  
 <!--卡片与列表整合-->  
 <div class="list card">  
 <a class="item item-icon-right">  
 <div class="icon ion-happy"></div>  
 aaaaaaaaaaaa  
 </a>  
 <a class="item item-icon-right">  
 <div class="icon ion-happy"></div>  
 aaaaaaaaaaaa  
 </a>  
 <div class="item item-avatar">  
 <img src="images/Dior02.jpg">  
 <h2>aaaa</h2>  
 </div>  
 <div class="item item-body">  
 <img src="images/Dior01.jpg">  
 <p>sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss</p>  
 </div>  
 </div>  
</div>

6、ionic 删格系统（网格grid系统）：采用css3的弹性盒子（Flex box）模型  
 主要思想让容器有能力改变其子元素的宽、高，甚至先后顺序  
 从而以最佳的方式填充可用空间  
总结：flex的主要目的是为了适应所有类型的显示设备  
<style>

.col-40{  
 -webik-flex:0 0 40%;  
 flex:0 0 40%;  
 }  
 .col-offset-5{  
 margin-left: 5%;  
 }  
</style>

<!--同等大小网格-->  
<div class="row">  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
</div>  
  
<!--指定列宽  
一行默认12个网格（超出也有效）  
ionic预置的列宽：  
col-10 col-20 col-25 col-33 col-50 col-67 col-75 col-80 col-90  
自定义列宽：用css3人为指定  
-->  
<div class="row">  
 <div class="col col-40"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
</div>  
  
<!--偏移量网格  
col-offset-20  
.col-offset-5{  
 margin-left: 5%;  
}  
-->  
<div class="row">  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col col-offset-20"><img src="../images/Dior05.jpg"></div>  
 <div class="col col-offset-5"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
</div>  
  
<!--纵向对齐网格  
col-top 元素顶对齐（默认）  
col-center 元素居中对齐  
col-bottom 元素底对齐  
  
row-top 整行顶对齐（默认）  
row-center 整行居中  
row-bottom 整行底对齐  
-->  
<div class="row">  
 <div class="col col-40"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col col-center"><img src="../images/Dior05.jpg"></div>  
 <div class="col col-bottom"><img src="../images/Dior05.jpg"></div>  
</div>  
  
<div class="row row-bottom">  
 <div class="col col-40"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
</div>  
  
<!--响应式参数  
responsive-lg 小于平板横屏  
responsive-sm 小于手机横屏  
responsive-md 小于平板的竖屏  
利用css3的media 响应式设计  
-->  
<div class="row responsive-lg">  
 <div class="col col-40"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
 <div class="col"><img src="../images/Dior05.jpg"></div>  
</div>  
  
<!--选项卡 ionic tab  
tabs-icon-only 图标选项卡，只有图标无文字  
tabs-icon-top 图标置顶  
tabs-icon-left 图标置左  
  
tabs-top 选项卡置顶  
  
条纹式选项卡  
tabs-striped  
tabs-background-颜色值  
tabs-color-颜色值  
-->  
<div class="tabs tabs-assertive">  
 <a class="tab-item"><i class="icon ion-home">home</i></a>  
 <a class="tab-item"><i class="icon ion-android-settings">set</i></a>  
 <a class="tab-item"><i class="icon ion-social-snapchat">video</i></a>  
</div>  
  
<div class="tabs-striped tabs-top tabs-background-assertive tabs-color-balanced">  
 <div class="tabs tabs-icon-top">  
 <a class="tab-item active"><i class="icon ion-home"></i>home</a>  
 <a class="tab-item"><i class="icon ion-android-settings"></i>set</a>  
 <a class="tab-item"><i class="icon ion-social-snapchat"></i>video</a>  
 </div>  
</div>

7、

<!--ionic 表单控件-->  
<!--list-inset 内嵌表单-->  
<div class="list list-inset">  
 <label class="item item-input">  
 <!--input-label 输入框属性-->  
 <span class="input-label">username:</span>  
 <input type="text" placeholder="please input username">  
 </label>  
  
 <!--<!--堆叠标签 item-stacked-label 描述性标签占单独一行-->-->  
 <label class="item item-input item-stacked-label">  
 <span class="input-label">password:</span>  
 <input type="password">  
 </label>  
  
 <!--<!--浮动标签 item-floating-label-->-->  
 <!--<!--内嵌输入域 item-input-inset-->-->  
 <label class="item item-input item-floating-label">  
 <label class="item item-input item-input-inset">  
 <span class="input-label">password:</span>  
 <input type="password">  
 </label>  
   
 <!--复选框-->  
 <div class="item item-checkbox checkbox-assertive">  
 <label class="checkbox">  
 <input type="checkbox">  
 </label>  
 aaaaa  
 </div>  
   
 <!--单选框???-->  
 <div class="item item-radio">  
 <input type="radio" name="r1" value="r1">  
 <div class="item-content">aaaaa</div>  
 <i class="radio-icon ion-checkmark"></i>  
 </div>  
   
 <!--切换开关-->  
 <div class="item item-toggle">  
 aaaa  
 <label class="toggle toggle-assertive">  
 <input type="checkbox">  
 <div class="track">  
 <div class="handle"></div>  
 </div>  
 </label>  
 </div>  
 <div class="item item-toggle">  
 bbbb  
 <label class="toggle">  
 <input type="checkbox">  
 <div class="track">  
 <div class="handle"></div>  
 </div>  
 </label>  
 </div>  
   
 <!--选择 select-->  
 <div class="item item-input item-select">  
 <div class="input-label">selectmmm:</div>  
 <select>  
 <option>aaaa</option>  
 <option>bbbb</option>  
 <option>cccc</option>  
 </select>  
 </div>  
   
 <!--滑块range-->  
 <div class="item range">  
 <i class="icon ion-volume-low"></i>  
 <input type="range" name="volume">  
 <i class="icon ion-volume-high"></i>  
 </div>

8、

<script src="../js/angular.min.js"></script>

/\*  
\* angularjs  
\* 1、基于js开发的客户端的应用框架  
\* 2、2009年产生  
\* 3、angularjs适合做前后端的分离式开发，前后端交换频繁的项目  
\* 不适合做特效、动画等  
\*  
\* angularjs特性：  
\* MVC模式  
\* 模块系统  
\* 指令系统  
\* 依赖注入  
\* 双向数据绑定  
\*  
\* ng-app：该指令告诉angularjs该元素是angularjs的所有者（即需要angularjs来解析）  
\* ng-model：该指令将输入域的值绑定给应用程序的某个变量  
\* ng-bind：将应用程序变量绑定到html元素上  
\* ng-init：初始化  
\*  
\* angular 表达式  
\* {{}}---可以包含文字、运算符变量  
\* \*/

uname:<input type="text" ng-model="myname" ng-init="myname=[2,3,4]"><br>  
uname:<input type="text" ng-model="myname" ng-init="a=23;b=10"><br>  
  
uname:<input type="text" ng-model="myname" ng-init="users={name:'zs',age:23}"><br>  
显示：{{myname}}<br>  
<div ng-bind="myname"></div><br>  
{{3+4}}<br>  
{{5\*6}}<br>  
{{myname+7}}<br>  
{{myname[1]}}<br>  
{{a\*b}}<br>  
{{users.name+" "+users.age}}<br>

9、

<script src="../js/angular.min.js"></script>

<script>  
 /\*  
 \* angularjs控制器--MVC  
 \*  
 \* ng-controller 定义应用程序控制器  
 \* $scope 局部作用域  
 \* $rootScope 全局作用域  
 \* 作用域的查找顺序：先找局部作用域，如果没有，再找全局作用域  
 \* \*/  
 //1  
 var app=angular.module("myapp",[]);//模型 model  
 app.controller("myctrl",function($scope){//控制器 controller  
 $scope.a1="zs";  
 $scope.b1="ls";  
 });  
  
 //2  
 //依赖注入  
 /\*  
 \* $scope：注入局部作用域对象  
 \* $rootScope：注入全局作用域对象  
 \* \*/  
 var app=angular.module("myapp",[]);//模型 model  
 app.controller("myctrl01",function($scope,$rootScope){//控制器 controller  
 $scope.name="zs";//控制器属性  
 $rootScope.sex="男";  
 $scope.sex="女";  
 $scope.fun1=function(){//控制器方法  
 return $scope.name+'aaaaaa';  
 }  
 });  
 app.controller("myctrl02",function($scope){//控制器 controller  
 $scope.age=23;  
 $scope.sex="女01";  
 });  
  
 //3  
 angular.module("myapp",[]).controller("myctrl03",function($scope){  
 $scope.persons=[{name:'zs',age:23},{name:'ls',age:34},{name:'ww',age:45}]  
 });  
  
 //4  
 angular.module("myapp",[])  
 .controller("myctrl03",function($scope){  
 $scope.persons=[{name:'zs',age:23},{name:'ls',age:34},{name:'ww',age:45}]  
 })  
 .controller("myctrl",function($scope){  
 $scope.mynamge1='zhang';  
 $scope.fun1=function(){  
 return $scope.mynamge1+" san";  
 }  
 });  
</script>

<!--1-->  
<div ng-app="myapp" ng-controller="myctrl">  
 <input type="text" ng-model="mynamge1"><br>  
 显示：{{mynamge1}}  
  
 <!--视图 view-->  
 {{a1}}<br>  
 {{b1}}<br>  
</div>

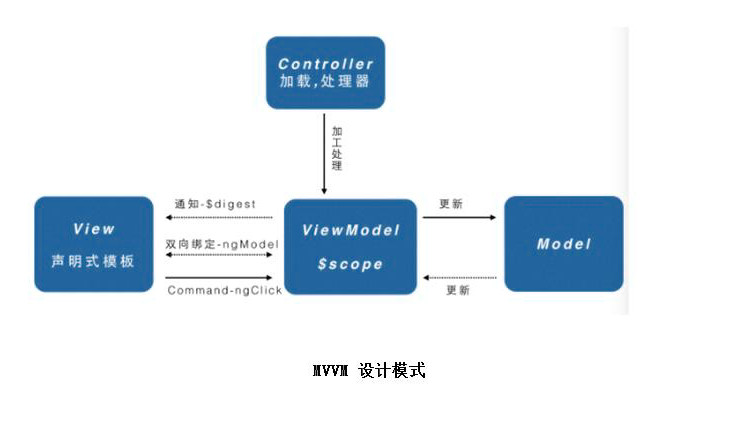
<!--2-->  
<div ng-controller="myctrl01">  
 {{name}}<br>  
 aa:{{sex}}<br>  
 bb:{{fun1()}}  
</div>  
<div ng-controller="myctrl02">  
 {{age}}<br>  
 aa:{{sex}}  
</div>

<!--3-->  
<div ng-controller="myctrl03">  
 <div ng-repeat="p in persons">  
 {{p.name+" "+p.age}}  
 </div>  
</div>

<!--4-->  
<div ng-controller="myctrl">  
 <input type="text" ng-model="mynamge1"><br>  
</div>  
姓：{{mynamge1}}

10、

10、



双向数据绑定MVVM  
\* Model(M):数据、数组、对象等存储着的内容  
\* View(V):视图，用户看到的部分  
\* ViewModel(VM):要将数据展现给用户，需要一定的逻辑，该层用来处理这部分逻辑  
\* $scope  
\* 双向：view改变的属性值，model会看到这个改变，而model改变，view也会看到  
\* 数据和表现相分离  
  
//2  
var app=angular.module("myApp",[]);  
//2、客户端改变model，该改变会自动在前端体现  
app.controller("myctrl",function($scope,$timeout){  
 $scope.myname="zs";  
 setTimeout(function(){$scope.myname="ww";},2000);//不会自动刷新页面  
 $timeout(function(){$scope.myname="ww";},2000);//自动刷新页面  
});  
  
//3  
app.controller("myctrl",function($scope,$timeout){  
 $scope.num=0;  
 $scope.add=function(w){  
 $scope.num+=w;  
 };  
 $scope.jian=function(e){  
 $scope.num-=e;  
 };  
  
 $scope.myname="zs";  
 /\*监听方法  
 \* $watch(arg1,arg2,arg3)  
 \* arg1:监听对象  
 \* arg2:当监听对象变化时，调用的函数或表达式  
 \* function(监听对象的新值，监听对象的旧值){}  
 \* arg3:是否深度监听 true--检测监听对象中每个属性的变化  
 \*  
 \* \*/  
 $scope.$watch("myname",function(newval,oldval){  
 alert(newval+" "+oldval);  
 },true);  
 $timeout(function(){$scope.myname="ww";},1000);  
  
 //4  
 $scope.users=[{name:"zs",age:23},{name:"ls",age:25},{name:"ww",age:43}]  
})

<body ng-app="myApp">  
<!--过滤器  
 currency 金额过滤器 总价：{{js()|currency:"￥"}}  
 number 保留小数点位数 总价：{{js()|number:2}}  
 uppercase 大写  
 lowercase 小写  
 orderBy 排序 orderBy:'name':true 降序  
 filter 过滤 按条件过滤  
-->

<!--1-->  
<div ng-app="">  
 <!--1、文本框中输入数据，下面的视图也会随之变化 数据绑定的一个方向 model-view-->  
 <input type="text" ng-model="myname"><br>  
 <h1>{{myname}}</h1>  
</div>

<!--2-->  
<div ng-app="myApp" ng-controller="myctrl">  
 <!--数据绑定的一个方向 model-view-->  
 <input type="text" ng-model="myname"><br>  
 <h1>{{myname}}</h1>  
</div>

<!--3-->  
<!--数据绑定可基于函数-->  
<div ng-controller="myctrl">  
 <button ng-click="add(3)">增加</button>  
 <button ng-click="jian(2)">减少</button><br>  
 当前数量：{{num}}  
 <input type="text" ng-model="myname">  
  
 <!--4-->  
 <input type="text" ng-model="gl">  
 <ul>  
 <li ng-repeat="u in users|filter:gl|orderBy:'name':true">  
 {{(u.name|uppercase)+" "+u.age}}  
 </li>  
 </ul>  
</div>

</body>

11、angularjs $http服务  
$http是angularjs的核心服务，可通过浏览器利用XMLHttpRequest对象或与jsonp与远程HTTP服务器进行交互

var app=angular.module("myApp",[]);  
app.controller("myctrl",function($scope,$http){  
 $scope.method01=function(){  
 $http({  
 method:"GET",  
 url:"test08\_php.php?uname="+$scope.myname  
 //params:{'uname':$scope.myname}  
 }).success(  
 function(data,status){  
 alert(data+" "+status);  
 }  
 ).error(  
 function(data,status){  
 alert(data+" "+status);  
 }  
 )  
 }  
});

<div ng-app="myApp" ng-controller="myctrl">  
 <input type="text" ng-model="myname">  
 <button ng-click="method01()">查询</button>  
</div>

PHP：test08\_php.php

<?php  
header("Content-type:text/html;charset=utf-8");  
$db\_link=mysql\_connect("localhost","root","123456") or die("数据库服务器连接失败".mysql\_error());  
$db=mysql\_select\_db("db43",$db\_link) or die("数据库连接失败".mysql\_error());  
$v\_uname=$\_REQUEST["uname"];  
$sql="select \* from t\_test where uname='$v\_uname'";  
mysql\_query("SET NAMES UTF8");  
$result=mysql\_query($sql,$db\_link);  
if($v1=mysql\_fetch\_array($result)){  
echo "success";  
}else{  
echo "false";  
}  
?>

12、

<script src="../js/model\_1.js"></script>

<script>

//模块应用

var app=angular.module("myApp",["myApp02","myApp03"]);//依赖模块

app.controller("myctrl01",function($scope){

$scope.myname="zs";

});

</script>

<body ng-app="myApp">

<div ng-controller="myctrl01">

{{myname}}

</div>

<div ng-controller="myctrl02">

{{myname}}

</div>

<div ng-controller="myctrl03">

{{myname}}

</div>

</body>

13、angularjs 路由 ng-include

<script src="../js/angular.min.js"></script>

<script src="../js/angular-route.min.js"></script>

//2

var app=angular.module("myApp",[]);

//3

var app=angular.module("myApp",[]);

app.controller("myctrl01",function($scope){

$scope.method=function(){

$scope.curTpl='test09.html';

};

});

//4

var app1=angular.module("myApp1",['ngRoute']);

//app1.config(function($routeProvider){});//简写

app1.config(["$routeProvider",function($routeProvider){

$routeProvider.when('/wj1',{

templateUrl:'test09.html',

controller:"myctr01"

}).when("/wj2",{

templateUrl:'test10.html'

}).otherwise({

//默认指回

redirectTo:"/wj1"

})

}]);

//定义控制器

app1.controller("myctr01",function($scope){

$scope.msg="....test....";

})

<div ng-app="myApp" ng-controller="myctrl01">

<!--1-->

<!--ng-include用字符串做解析，所以载套用一层单引号-->

<div ng-include="'a1.html'"></div>

<div ng-include src="'a1.html'"></div>

<!--2-->

<!--模板文件 适用于移动端-->

<script type="text/ng-template" id="test09.html">content of the template</script>

<!--触发加载-->

<a ng-click="curTpl='test09.html'">加载模板文件</a>

<div ng-include src="curTpl"></div>

<!--3-->

<script type="text/ng-template" id="test09.html">content of the template</script>

<a ng-click="method()">加载模板文件</a>

<div ng-include src="curTpl"></div>

</div>

<!--4、路由-->

<div ng-app="myApp1">

<a href="#wj1">文件1</a>

<a href="#wj2">文件2</a>

<div ng-view></div>

<script type="text/ng-template" id="test09.html">content of the {{msg}} template</script>

<script type="text/ng-template" id="test10.html">111111</script>

</div>

14、angularjs 服务 provider factory service

var app1=angular.module("myApp",[]);

//定义服务

app1.factory("myservice",function(){

var f={};

f.square=function(a){

return a\*a;

};

return f;

});

//应用服务

app1.controller("myctrl",function($scope,myservice){

$scope.m1=function(){

$scope.result=myservice.square($scope.num);

};

});

<div ng-app="myApp" ng-controller="myctrl">

<input type="text" ng-model="num">

<button ng-click="m1()">求平方</button>

<br>

{{result}}

</div>

15、

<html ng-app="myApp">

<meta name="viewport" content="initial-scale=1,user-scalable=no,width=device-width">

<link href="../css/ionic.min.css" rel="stylesheet">

<link href="../css/ionicons.min.css" rel="stylesheet">

<!--本质上是angularjs-->

<script src="../js/ionic.bundle.min.js"></script>

<script>

var app=angular.module("myApp",["ionic"]);

app.controller("myctrl",function($scope){

$scope.items=[];

for(var i=0;i<30;i++){

$scope.items.push("test"+i);

}

});

</script>

<body ng-controller="myctrl">

<!--头-->

<ion-header-bar class="bar-positive" align-title="center">

<h1 class="title">--header--</h1>

</ion-header-bar>

<!--主体-->

<ion-content>

<ul class="list">

<li class="item" ng-repeat="a in items">{{a}}</li>

</ul>

</ion-content>

<!--尾-->

<ion-footer-bar class="dark-bg">

<h1 class="title light">--footer--</h1>

</ion-footer-bar>

</body>

</html>

a1.html内容：

<span>aaaaaaaaaaa</span>

16、

var app=angular.module("myApp",["ionic"]);

<!--1-->  
<!--可触摸滚动  
ion-scroll：声明一个可滚动的容器元素，用户可按住内容进行拖动  
direction：表示内容可以滚动的方向 x\y\xy  
zooming：是否支持“捏拉缩放  
-->  
<ion-header-bar class="bar-positive">  
 <h1 class="title">header</h1>  
</ion-header-bar>  
<ion-scroll class="has-header" zooming="true" style="width: 400px;height:400px;overflow: hidden" direction="xy">  
 <div style="width:2000px;height: 2000px;" id="allmap"></div>  
</ion-scroll>

//2  
app.controller("myctrl",function($scope){  
 $scope.items=['test1','test2','test3'];  
 //动态添加数据  
 var b=4;  
 $scope.dorefresh=function(){  
 for(var i=0;i<10;i++,b++){  
 $scope.items.unshift("test"+b);  
 }  
 //刷新完毕后通知框架  
 $scope.$broadcast("scroll.refreshComplete")  
 }  
});

<!--2-->  
<!--下拉刷新  
ion-refresher  
on-refresh：用户下拉足够距离时执行的函数或表达式  
on-pulling：用户下拉时执行  
pulling-text：用户下拉时显示的文本  
pulling-icon：用户下拉时显示的图标  
refreshing-icon：用户下拉并松开后的等待图标 现用户spinner代替  
-->  
<ion-header-bar class="bar-positive">  
 <h1 class="title">header</h1>  
</ion-header-bar>  
<ion-content>  
 <ion-refresher on-refresh="dorefresh()" pulling-text="pull refresh...." spinner="android"></ion-refresher>  
 <ul class="list">  
 <li class="item" ng-repeat="i in items">{{i}}</li>  
 </ul>  
</ion-content>  
  
//3  
var b=4;  
app.controller("myctrl",function($scope,$timeout){  
 $scope.items=['test1','test2','test3'];  
 $scope.load\_infin=function(){  
 $timeout(function(){  
 for(var i=0;i<3;i++,b++){  
 $scope.items.unshift("test"+b);  
 }  
 //刷新完毕后通知框架  
 $scope.$broadcast("scroll.infiniteScrollComplete")  
 },1000);  
 }  
});

<ion-header-bar class="bar-positive">  
 <h1 class="title">header</h1>  
</ion-header-bar>  
<ion-content>  
 <ion-infinite-scroll icon="ion-load-a" distance="1%" immediate-check="true" on-infinite="load\_infin()"></ion-infinite-scroll>  
 <ul class="list">  
 <li class="item" ng-repeat="i in items">{{i}}</li>  
 </ul>  
</ion-content>  
  
//4  
app.controller("myctrl",function($scope,$ionicScrollDelegate){  
 $ionicScrollDelegate.scrollTop(true);  
});

<!--4-->  
<!--  
$ionicScrollDelegate指令 控制滚动容器  
提供的常用方法如下：  
scrollTop(arg)：滚动到内容顶部  
 arg：true\false 是否使用动画展示滚动过程  
scrollBottom(arg)：滚动到内容底部  
 参数同上  
scrollTo(arg1，arg2)：滚动到指定位置  
 arg1：滚动到的x坐标  
 arg2：滚动到的y坐标  
scrollBy(arg1，arg2)：滚动指定偏移量  
 arg1：滚动到的x偏移量  
 arg2：滚动到的y偏移量  
resize()：重新计算容器尺寸  
-->

17、百度地图API功能

<!DOCTYPE html>  
<html>  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
 <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />  
 <style type="text/css">  
 body, html,#allmap {width: 100%;height: 100%;overflow: hidden;margin:0;font-family:"微软雅黑";}  
 </style>  
 <script type="text/javascript" src="http://api.map.baidu.com/api?v=2.0&ak=ehfpBLLcGQD9hYPBYcza5wHoGPkOWh3e"></script>  
 <title>根据城市名设置地图中心点</title>  
</head>  
<body>  
<div id="allmap"></div>  
</body>  
</html>  
<script type="text/javascript">  
 // 百度地图API功能  
 var map = new BMap.Map("allmap"); // 创建Map实例  
 map.centerAndZoom("哈尔滨",15); // 初始化地图,用城市名设置地图中心点  
 map.enableScrollWheelZoom(true);//可以抓取  
</script>

18、

<script>  
 var app=angular.module("myApp",["ionic"]);  
 app.controller("myctrl",function($scope,$ionicListDelegate,$ionicPopup){  
 $scope.item=["test1","test2","test3","test4","test5"];  
 $scope.myclose=function(){  
 $ionicListDelegate.closeOptionButtons();  
 };  
 $scope.flag={showDelete:false};  
 $scope.mydelete=function(a){  
 var con=$ionicPopup.confirm({  
 title:"询问",  
 template:"是否删除?"  
 });  
 con.then(function(r){  
 if(r){  
 var id = $scope.item.indexOf(a);  
 $scope.item.splice(id, 1);  
 }else{  
  
 }  
 });  
 };  
 $scope.myorder=function(i,fromIndex,toIndex){  
 $scope.item.splice(fromIndex,1);  
 $scope.item.splice(toIndex,0,i);  
 };  
 });  
</script>

<body ng-controller="myctrl">  
<!--列表  
预定义按钮  
ion-option-button 选项按钮  
ion-delete-button 删除按钮  
ion-recoder-button 重排按钮  
  
$ionicListDelegate 由脚本控制列表元素  
 $ionicListDelegate.closeOptionButtons() 关闭所有选项按钮  
  
show-delete:是否显示成员内删除按钮  
show-reorder 是否显示成员的reorder按钮  
-->  
<ion-header-bar class="bar-positive" align-title="center">  
 <a class="button button-clear icon ion-ios-minus-outline" ng-click="flag.showDelete=!flag.showDelete"></a>  
 <h1 class="title">head</h1>  
 <!--<a class="button" ng-click="myclose()">关闭选项</a>-->  
 <button class="button" ng-click="flag.showReorder=!flag.showReorder">reorder</button>  
</ion-header-bar>  
<ion-content class="has-header">  
 <ion-list show-delete="flag.showDelete" show-reorder="flag.showReorder">  
 <ion-item ng-repeat="a in item">  
 {{a}}  
 <ion-option-button class="button-calm icon ion-home"></ion-option-button>  
 <ion-option-button class="button-assertive" ng-click="mymodify(i)">modify</ion-option-button>  
 <ion-delete-button class="icon ion-minus-circled" ng-click="mydelete(a)"></ion-delete-button>  
 <ion-reorder-button class="ion-navicon" on-reorder="myorder(i,$fromIndex,$toIndex)"></ion-reorder-button>  
 </ion-item>  
 </ion-list>  
</ion-content>  
</body>

19、对话框---$ionicPopup  
angular.module("myApp",["ionic"])  
 .controller("myctrl",function($scope,$ionicPopup,$timer){  
 $scope.showconfirm=function(){  
 var con=$ionicPopup.confirm({  
 title:"询问",  
 template:"你是谁?"  
 });  
 con.then(function(r){  
 if(r){  
 $scope.msg="是";  
 }else{  
 $scope.msg="否";  
 }  
 })  
 };  
 $scope.showalert=function(){  
 $ionicPopup.alert({  
 title:"提示",  
 template:"<span style='color:red'>登录成功！</span>"  
 });  
 };  
 $scope.showprompt=function(){  
 var con=$ionicPopup.prompt({  
 title:"提示",  
 template:"你多大了？"  
 });  
 con.then(function(r){  
 if(isNaN(r)){  
 alert("请输入数字");  
 }else{  
 if(r>20){  
 alert("大于20岁");  
 }else if(r>100){  
 alert("大于100岁");  
 }  
 }  
  
 })  
 };  
 $scope.mydialog=function(){  
 $scope.mm={};  
 var show=$ionicPopup.show({  
 title:"你的银行卡密码是多少？",  
 template:"<input type='password' ng-model='mm.yhk\_pwd'>",  
 subTitle:"你放心我不告诉别人",  
 scope:$scope,  
 buttons:[{text:'取消'},{text:'确定',onTap:function(){  
 return $scope.mm.yhk\_pwd;  
 }}]  
 });  
 show.then(function(res){  
 $scope.msg=res;  
 });  
 };  
 })  
 .controller("myctrl",function($scope,$ionicActionSheet){  
 $scope.mydialog02=function(){  
 $ionicActionSheet.show({  
 titleText:"操作当前内容",  
 buttons:[{text:'拷贝'},{text:'粘贴'},{text:'移动'}],  
 cancelText:"取消",  
 cancel:function(){  
 alert("dddd");  
 }  
 });  
  
 }  
 })  
 .controller("myctrl",function($scope,$ionicPopover){  
 // $scope.pover=$ionicPopover.fromTemplateUrl("fd\_test.html",{scope:$scope});  
 $ionicPopover.fromTemplateUrl("fd\_test.html",{scope:$scope})  
 .then(function(p){$scope.popver=p;});  
 $scope.mydialog03=function($event){  
 $scope.popver.show($event);  
 }  
 });

<body ng-controller="myctrl">  
<div>{{msg}}</div>  
<button class="button button-assertive" ng-click="showconfirm()">确认对话框</button>  
<button class="button button-positive" ng-click="showalert()">警告（消息）对话框</button>  
<button class="button button-dark" ng-click="showprompt()">输入对话框</button>  
<button class="button button-positive" ng-click="mydialog()">自定义对话框</button>  
<button class="button button-positive" ng-click="mydialog02()">下弹对话框</button>  
<button class="button button-positive" ng-click="mydialog03($event)">浮动窗口</button>  
<script type="text/ng-template" id="fd\_test.html">  
<ion-popover-view>  
 <ion-header-bar>  
 <h1 class="title">header</h1>  
 </ion-header-bar>  
 <ion-content>  
 aaaaaaaaaaaaaaaaaaaaaaa  
 </ion-content>  
</ion-popover-view>  
</script>  
</body>

20、

//ionic 路由  
var app=angular.module("myApp",["ionic"]);  
//1  
app.controller("myctrl",function($scope,$http,$templateCache){  
 $scope.load=function(){  
 //模板缓存  
 $http.get("n1\_tmp",{cache:$templateCache})  
 .success(function(data,staus){  
 //var v1=document.querySelector("#mycontent");  
 //var v1=document.querySelectorAll(".aa");//集合  
 var v1=document.getElementById("mycontent");  
 angular.element(v1).html(data)  
 })  
 .error(function(data,staus){  
  
 });  
 }  
});  
  
//2  
//路由配置  
app.config(function($stateProvider,$urlRouterProvider){  
 $stateProvider.state("zy",{templateUrl:"../templates/home.html"})//内联模板id  
 .state("news",{  
 url:"/xw",//超链接href=“#xw”  
 templateUrl:"../templates/mynews.html",  
 controller:"mynewctrl"  
 })  
});  
app.controller("myctrl",function($scope,$state){  
 $state.go("zy");  
});  
app.controller("mynewctrl",function($scope){  
 alert("aaaa");  
});

<!--1-->  
<ion-header-bar class="bar-positive">  
 <h1 class="title">head</h1>  
 <button class="button button-assertive" ng-click="load()">加载</button>  
</ion-header-bar>  
<ion-content>  
 <div id="mycontent"></div>  
</ion-content>  
<!--内联模板-->  
<script type="text/ng-template" id="n1\_tmp">  
<div>dddddddddddddd</div>  
</script>

<!--2-->  
<!--视图容器-->  
<div ui-view=""></div>  
<!--导航视图 等价于 视图容器-->  
<ion-nav-view animation="silde-in-up"></ion-nav-view>  
  
<!--主页-->  
<script type="text/ng-template" id="home.html">  
<ion-header-bar class="bar-positive">  
 <h1 class="title">主页</h1>  
</ion-header-bar>  
<ion-content>  
 <ul class="list list-inset">  
 <!--ui-sref 该指令指向state路由-->  
 <li class="item" ui-sref="news">新闻</li>  
 <li class="item">娱乐</li>  
 <li class="item">体育</li>  
 </ul>  
</ion-content>  
</script>  
<!--分页--新闻-->  
<script type="text/ng-template" id="mynews.html">  
 <ion-header-bar class="bar-positive">  
 <h1 class="title">新闻</h1>  
 </ion-header-bar>  
 <ion-content>  
 <ul class="list list-inset">  
 <!--ui-sref 该指令指向state路由-->  
 <li class="item" ui-sref="zy">主页</li>  
 <li class="item">娱乐</li>  
 <li class="item">体育</li>  
 </ul>  
 </ion-content>  
</script>

21、选项卡tab

<ion-header-bar class="bar-positive">  
 <h1 class="title">header</h1>  
</ion-header-bar>  
<!--选项卡tab  
tabs-striped 选中有白线  
tabs-top 选项卡在上部（默认下部）  
tabs-icon-top 图标在上面，文字在下  
icon-on 选中时图标样式  
icon-off 非选中时图标样式  
-->  
<ion-tabs class="tabs-positive tabs-striped tabs-icon-top">  
 <ion-tab title="tab1" icon-on="ion-ios-clock" icon-off="ion-ios-clock-outline">  
 <ion-view>  
 <ion-content class="calm-bg">tab11111111</ion-content>  
 </ion-view>  
 </ion-tab>  
 <ion-tab title="tab2" icon-on="ion-ios-clock" icon-off="ion-ios-clock-outline">  
 <ion-view>  
 <ion-content class="calm-border">tab22222222</ion-content>  
 </ion-view>  
 </ion-tab>  
 <ion-tab title="tab3" icon-on="ion-ios-clock" icon-off="ion-ios-clock-outline">  
 <ion-view>  
 <ion-content class="calm-bg">tab33333333</ion-content>  
 </ion-view>  
 </ion-tab>  
</ion-tabs>

22、

<script>  
 var app=angular.module("myApp",["ionic"]);  
 app.config(function($stateProvider,$urlRouterProvider){  
 $stateProvider.state("news",  
 {  
 url:"/news",  
 abstract:true,  
 templateUrl:"templates/news.html"  
 })  
 .state("news.content",  
 {  
 url:"/content",  
 views:{"new\_con1":

{templateUrl:"templates/news\_content1.html"}}  
 });  
 $urlRouterProvider.otherwise("/news/content");  
 });  
</script>

<body>  
<ion-nav-view></ion-nav-view>  
</body>

news.html内容：

<ion-tabs class="tabs-assertive tabs-icon-top">  
 <ion-tab title="内容1" icon-on="ion-ios-pulse"

icon-off="ion-ios-pulse-strong">  
 <ion-nav-view name="new\_con1"></ion-nav-view>  
 </ion-tab>  
 <ion-tab title="内容2" icon-on="ion-ios-pulse"

icon-off="ion-ios-pulse-strong">  
 <ion-nav-view name="new\_con2"></ion-nav-view>  
 </ion-tab>  
</ion-tabs>

news\_content1.html内容：

<ion-view view-title="news content">  
 <ion-content class="has-header">  
 new content1111111111111111  
 </ion-content>  
</ion-view>

1. 侧栏 menu-toggle="left" menu-toggle="right" 侧栏菜单切换  
   <ion-side-menus>  
    <ion-side-menu side="left" style="width: 150px;" class="balanced-bg">我是左

面...</ion-side-menu>  
 <ion-side-menu-content class="calm-bg">  
 <!--<div class="bar bar-header bar-positive">-->  
 <!--<h1>...test...</h1>-->  
 <!--</div>-->   
  
 <ion-header-bar class="bar-positive">  
 <a class="button icon ion-navicon" menu-toggle="left"></a>  
 <h1 class="title">...test...</h1>  
 <a class="button icon ion-navicon" menu-toggle="right"></a>  
 </ion-header-bar>  
 </ion-side-menu-content>

<ion-side-menu side="right" style="width: 150px;" class="dark-bg">我是右

面...</ion-side-menu>  
</ion-side-menus>

24、留言板

window.onload=function(){

document.getElementById("ly").addEventListener("click",function(){  
 var v1=document.getElementById("bt").value;  
 var v2=document.getElementById("nr").value;  
 var str="标题"+v1+"内容"+v2;  
 localStorage.setItem(localStorage.length,str);  
 },false);  
 document.getElementById("cxly").addEventListener("click",function(){  
 var str1="";  
 for(var i=0;i<localStorage.length;i++){  
 var k=localStorage.key(i);  
 str1=str1+localStorage.getItem(k)+"<br>";  
 }  
 document.getElementById("show").innerHTML=str1;  
 },false);  
}

标题:<input type="text" id="bt"><br><br>  
内容:<textarea id="nr"></textarea><br><br>  
<button id="ly">留言</button>  
<button id="cxly">查看所有留言</button><br><br>  
<div id="show"></div>

25、html5 本地存储 web storage  
\* sessionStorage：临时保存  
\* sessionStorage.setItem(key value); //保存数据  
\* sessionStorage.getItem(key) //提取数据  
\* 将数据保存在浏览器的session对象中，如果浏览器关闭保存数据消失  
\*  
\* localStorage:永久保存  
\* localStorage.setItem(key value); //保存数据  
\* localStorage.getItem(key) //提取数据  
\* localStorage.removeItem(key) //删除数据  
\* 将数据保存在客户端的本地硬件设备上（一般指硬盘）  
\* 如浏览器关闭，数据依然存在  
\* localStorage没有过期设置，需要人为定时器设置删除  
\*  
\* 共同属性方法：  
\* localStorage.length (条目数量)  
\* sessionStorage.length  
\*  
\* 提取条目(根据索引值（下标0开始）返回key值)  
\* var k=localStorage.key(0) //key值  
\* localStorage.getItem(k) //key的value值  
\*  
\* 清空所有数据  
\* localStorage.clear()  
\*  
\* json格式数据  
\* {a1:"23",b1:"34"....}  
\*  
\* JSON.parse 将字符串解析为json格式对象  
\* JSON.stringify 将一个实体对象转换为json格式的文本数据（对象转换为字符串）

//json格式的存储于提取  
var person=new Object();  
person.name='zs';  
person.age=23;  
//var person={name:'zs',age:23}  
  
//json存数据  
var s=JSON.stringify(person);  
localStorage.setItem("bb",s);  
  
//json取数据  
var str=localStorage.getItem("bb");  
var s=JSON.parse(str);  
alert(s.name+" "+ s.age);

window.onload=function(){  
 document.getElementById("b1").addEventListener("click",function(){  
 var v1=document.getElementById("inp01").value;  
 sessionStorage.setItem("aa",v1); //key value  
 },false);  
 document.getElementById("b2").addEventListener("click",function(){  
 var v2=sessionStorage.getItem("aa"); //读取  
 alert(v2)  
 },false);

document.getElementById("b3").addEventListener("click",function(){  
 var v1=document.getElementById("inp01").value;  
 localStorage.setItem("bb",v1); //key value  
 },false);  
 document.getElementById("b4").addEventListener("click",function(){  
 var v2=localStorage.getItem("bb"); //读取  
 alert(v2)  
 },false);  
}

<input type="text" id="inp01"><br><br>  
<button id="b1">sessionStroage保存数据</button>  
<button id="b2">提取sessionStroage数据</button><br><br>  
  
<button id="b3">LocalStorage保存数据</button>  
<button id="b4">提取LocalStorage数据</button>

26、js cookie  
\* 1、cookie可能会被禁用  
\* 2、cookie存储在客户端（硬盘）  
\* 3、cookie可能被清除（杀毒软件或人为）  
\* 4、安全性不高  
\* 5、cookie可以设置过期时间  
  
//设置cookie  
document.cookie="userid=001;pwd=123";  
document.cookie="userid=001;";  
document.cookie="pwd=123;";  
  
//cookie不允许出现 "" = & 等特殊符号,可进行编码  
//加密与解密  
//escape 加密 将符号用16进制表示，如 空格20%  
//unescape 解密（解码）  
document.cookie="str="+escape('my name is=aa');  
  
//提取  
var v1=document.cookie;  
var arr1=v1.split(";");  
for(var i=0;i<arr1.length;i++){  
 var arr2=arr1[i].split("=");  
 document.write("Name="+arr2[0]+"，Value="+arr2[1]+"<br>");  
}  
  
//过期设置 expires  
var d=new Date();  
//1分钟后过期  
d.setTime(d.getTime()+1000);  
//10天后过期  
d.setDate(d.getDate()+10);  
document.cookie="myname='zs';expires="+ d.toGMTString();  
  
//删除cookie--指定过期时间

1. html5 本地数据库  
    \* 将数据存储在客户端，采用“sqllite文件类型sql数据库”  
    \* 1、经常用到的数据会保存在本地数据库中，不用去服务器请求，提高效率  
    \* 2、离线应用，如写日记，离线时存储到本地数据库，在线时上传  
    \* \*/  
    //打开连接并创建数据库  
    //openDatabase（数据库名，版本号，数据库描述，数据库大小，回调（可省略））  
    /\*  
    \* 事务transaction：可以防止在对数据库进行访问及执行有关操作时受到外界干

\* 扰，即使用事务来达到在操作完成之前，阻止别的用户访问数据库的目的  
 \* \*/  
 var mydb=openDatabase("testdb","1.0","测试本地数据库",1000000);  
 if(mydb){  
 alert("本地数据库创建成功！");  
 }else{  
 alert("本地数据库创建失败！");  
 }  
 //创建表  
 /\*  
 \* ts.executeSql(arg1,arg2,arg3,arg4)  
 \* arg1：要执行的sql语句  
 \* arg2：插入到sql语句中问号所在处的字符串数据  
 \* arg3：成功回调函数  
 \* arg4：失败回调函数  
 \* arg2,arg3,arg4可省略  
 \* \*/  
 mydb.transaction(function(ts){  
 ts.executeSql("create table if not exists users(id integer

unique,username varchar(10),pwd varchar(8))",  
 [],  
 function(){alert("创建表成功");},  
 function(){alert("创建表失败");});  
 });  
 //插入数据  
 mydb.transaction(function(ts){  
 //ts.executeSql("insert into users(id,username,pwd)

values(1,'zs','123')");  
 ts.executeSql("insert into users(id,username,pwd)

values(?,?,?)",[2,'ls','123']);  
 });  
  
 //批量插入  
 //[['zs',34],['ls',34],['ww',34],['gg',33]]  
 var mydb=openDatabase("testdb01","1.0","测试本地数据库",1000000);  
 if(mydb){  
 alert("本地数据库创建成功！");  
 }else{  
 alert("本地数据库创建失败！");  
 }  
 mydb.transaction(function(ts){  
 ts.executeSql("create table if not exists user(username

varchar(10),pwd varchar(8))",  
 [],  
 function(){alert("创建表成功");},  
 function(){alert("创建表失败");});  
 });  
 var arr=[['zs',34],['ls',34],['ww',34],['gg',33]];  
 mydb.transaction(function(ts){  
 for(var i=0;i<arr.length;i++){  
 ts.executeSql("insert into user(username,pwd)

values(?,?)",arr[i]);  
 }  
 });

28、查询  
mydb.transaction(function(tx){  
 //1  
 tx.executeSql("select \* from users where id=?",[1],function(tx,resultset){  
 //遍历结果集  
 for(var i=0;i<resultset.rows.length;i++){  
 alert(resultset.rows.item(i).username);  
 alert(resultset.rows.item(i).pwd);  
 }  
 });  
  
 //2  
 tx.executeSql("select \* from users",[],function(tx,resultset){  
 //遍历结果集  
 var a="";  
 for(var i=0;i<resultset.rows.length;i++){  
 a+=resultset.rows.item(i).id+" "+resultset.rows.item(i).username+" "+resultset.rows.item(i).pwd+" "+"<a onclick='delfun("+resultset.rows.item(i).id+")'>删除</a>"+"<br>"  
 }  
 document.getElementById("show").innerHTML=a;  
 });  
});

删除函数  
function delfun(id){  
 mydb.transaction(function(tc){  
 tc.executeSql("delete from users where id=?",[id]);  
 });  
 mydb.transaction(function(tx){  
 tx.executeSql("select \* from users",[],function(tx,resultset){  
 //遍历结果集  
 var a="";  
 for(var i=0;i<resultset.rows.length;i++){  
 a+=resultset.rows.item(i).id+" "+resultset.rows.item(i).username+" "+resultset.rows.item(i).pwd+" "+"<a onclick='delfun("+resultset.rows.item(i).id+")'>删除</a>"+"<br>"  
 }  
 document.getElementById("show").innerHTML=a;  
 });  
 });  
}

<div id="show"></div>

29、tab页内容切换

var app=angular.module("myApp",["ionic"]);  
app.config(function($stateProvider,$urlRouterProvider){  
 $stateProvider.state("menuhome",{  
 url:"/myside",  
 abstract:true,  
 templateUrl:"templates/test25\_side.html"  
 })  
 .state("menuhome.content1",{url:"/side1",  
 views:{"menucontent":{  
 templateUrl:"myProject/templates/test25\_side01.html" }  
  
 }})  
 .state("menuhome.content2",{url:"/side2",  
 views:{"menucontent":{  
 templateUrl:"templates/test25\_side02.html" }  
  
 }})  
 .state("menuhome.content3",{url:"/side3",  
 views:{"menucontent":{  
 templateUrl:"templates/test25\_side03.html" }  
  
 }});  
 $urlRouterProvider.otherwise("/myside/side1");  
});

<body>  
<ion-nav-view></ion-nav-view>  
</body>

test25\_side.html内容：

<ion-side-menus>  
 <ion-side-menu side="left">  
 <div class="bar bar-header bar-stable">  
 <h1 class="title">side\_left</h1>  
 </div>  
 <ion-content class="has-header">  
 <ion-list>  
 <ion-item href="#myside/side1">item111</ion-item>  
 <ion-item href="#myside/side2">item222</ion-item>  
 <ion-item href="#myside/side3">item333</ion-item>  
 </ion-list>  
 </ion-content>  
 </ion-side-menu>  
  
 <ion-side-menu-content class="calm-bg">  
 <ion-nav-bar class="bar-stable">  
 <ion-nav-back-button class="button-clear">回退

</ion-nav-back-button>  
 </ion-nav-bar>  
 <ion-nav-view name="menucontent" animation="slide-left-right"></ion-nav-view>  
 </ion-side-menu-content>  
</ion-side-menus>

test25\_side01.html内容：

<ion-view>  
 <ion-nav-buttons>  
 <button menu-toggle="left" class="button button-icon

ion-navicon"></button>  
 </ion-nav-buttons>  
 <ion-content class="has-header">  
 aaaaa1111111111111111  
 </ion-content>  
</ion-view>

30、html5音频处理  
\* 一、属性  
\* src:指定媒体数据的url地址  
\* autoplay:媒体是否在页面加载完后自动播放  
\* true  
\* false  
\* loop:是否循环播放媒体  
\* true  
\* false  
\* played:返回布尔型，表示当前媒体处于播放状态  
\* paused:返回布尔型，表示当前媒体处于暂停状态  
\* ended:表示当前媒体是否播放完毕  
\*  
\* playbackRate:读取或修改媒体当前的播放速率（播放倍数）  
\* 默认速率1  
\*  
\* volume:读取或修改媒体的播放音量  
\* 范围：0-1 0-静音 1-最大  
\*  
\* muted:读取或修改媒体的静音状态  
\* true--静音状态  
\* false--非静音状态  
\*  
\* duration:媒体文件的总播放时间  
\* currentTime:媒体文件播放的当前时间  
\* 单位：秒  
\* 二、方法  
\* play() 播放  
\* pause() 暂停  
\* load() 重新加载  
\* 三、事件  
\* timeupdate:当前播放位置被改变触发该事件  
\* ended:媒体播放结束事件

window.onload=function(){  
 //动态加载音频  
 var au=document.createElement("audio");  
 au.setAttribute("autoplay",true);  
 au.setAttribute("loop",true);  
 au.setAttribute("src","../media/123.mp3");  
 document.body.appendChild(au);  
  
 //进度条控制  
 var au01=document.getElementById("au01");  
 var au02=document.getElementById("pro\_id");  
 document.getElementById("au01").addEventListener("timeupdate",fun1,false);  
 function fun1(){  
 var v1,v2,v3,v4;  
 if(Math.floor(au01.currentTime/60)<10){  
 v1="0"+Math.floor(au01.currentTime/60);  
 }else{  
 v1=Math.floor(au01.currentTime/60);  
 }  
 if(Math.floor(au01.currentTime%60)<10){  
 v2="0"+Math.floor(au01.currentTime%60);  
 }else{  
 v2=Math.floor(au01.currentTime%60);  
 }  
 if(Math.floor(au01.duration/60)<10){  
 v3="0"+Math.floor(au01.duration/60);  
 }else{  
 v3=Math.floor(au01.duration/60);  
 }  
 if(Math.floor(au01.duration%60)<10){  
 v4="0"+Math.floor(au01.duration%60);  
 }else{  
 v4=Math.floor(au01.duration%60);  
 }  
 document.getElementById("show").innerHTML=v1+":"+v2+"|"+v3+":"+v4;  
 au02.setAttribute("value",au01.currentTime/au01.duration\*100);  
 }  
  
};  
function fun7(e){  
 var au01=document.getElementById("au01");  
 var v=document.getElementById("pro\_id");  
 au01.currentTime=Math.floor((e.offsetX/v.clientWidthnt)\*au01.duration);  
  
}  
  
var au01=document.getElementById("au01");  
var arr1=["../media/123.mp3","../media/456.mp3","../media/123.mp3"];  
var i=0;  
document.getElementById("au01").addEventListener("ended",function(){  
 if(au01.ended){  
 i++;  
 document.getElementById("au01").setAttribute("src",arr1[i]);  
 if(i+1==3){  
 i=0;  
 }  
 }  
},false);  
function fun8(){  
 document.getElementById("au01").setAttribute("src",arr1[i]);  
 i++;  
 if(i==3){  
 i=0;  
 }  
}  
function fun9(){  
 document.getElementById("au01").setAttribute("src",arr1[i]);  
 i--;  
 if(i==0){  
 i=2;  
 }  
}  
  
//封装  
var au01=document.getElementById("au01");  
var au\_control={  
 fun1:function(){  
 var but01=document.getElementById("but01");  
 if(au01.paused){  
 au01.play();  
 but01.innerText="暂停";  
 }else if(au01.played){  
 au01.pause();  
 but01.innerText="播放";  
 }  
 },  
 fun2:function(){  
 au01.load();  
 },  
 fun3:function(){  
 au01.playbackRate=4;  
 },  
 fun4:function(){  
 if(au01.volume<=1){  
 au01.volume=au01.volume+0.1;  
 }  
 },  
 fun5:function(){  
 if(au01.volume>0){  
 au01.volume=au01.volume-0.1;  
 }  
 },  
 fun6:function(){  
 var but06=document.getElementById("but06");  
 if(au01.muted){  
 au01.volume=1;  
 but06.innerText="非静音";  
 }else if(!au01.muted){  
 au01.volume=0;  
 but06.innerText="静音";  
 }  
 }  
};  
document.getElementById("but01").onclick=au\_control.fun1();  
document.getElementById("but02").onclick=au\_control.fun2();  
document.getElementById("but03").onclick=au\_control.fun3();  
document.getElementById("but04").onclick=au\_control.fun4();  
document.getElementById("but05").onclick=au\_control.fun5();  
document.getElementById("but06").onclick=au\_control.fun6();  
  
  
  
//播放、暂停切换  
function fun1(){  
 var au01=document.getElementById("au01");  
 var but01=document.getElementById("but01");  
 if(au01.paused){  
 au01.play();  
 but01.innerText="暂停";  
 }else if(au01.played){  
 au01.pause();  
 but01.innerText="播放";  
 }  
}  
//重放  
function fun2(){  
 var au01=document.getElementById("au01");  
 au01.load();  
}  
//快进  
function fun3(){  
 var au01=document.getElementById("au01");  
 au01.playbackRate=4;  
}  
//音量  
function fun4(){  
 var au01=document.getElementById("au01");  
 if(au01.volume<=1){  
 au01.volume=au01.volume+0.1;  
 }  
}  
function fun5(){  
 var au01=document.getElementById("au01");  
 if(au01.volume>0){  
 au01.volume=au01.volume-0.1;  
 }  
}  
//静音切换  
function fun6(){  
 var au01=document.getElementById("au01");  
 var but06=document.getElementById("but06");  
 if(au01.muted){  
 au01.volume=1;  
 but06.innerText="非静音";  
 }else if(!au01.muted){  
 au01.volume=0;  
 but06.innerText="静音";  
 }  
}

<body>  
<audio src="../media/123.mp3" autoplay loop controls id="au01"></audio>  
<video src="../media/123.mp4" autoplay loop controls></video>  
  
<button onclick="fun1()" id="but01">暂停</button>  
<button onclick="fun2()" id="but02">重放</button>  
<button onclick="fun3()" id="but03">快进</button>  
<button onclick="fun4()" id="but04">增大音量</button>  
<button onclick="fun5()" id="but05">减少音量</button>  
<button onclick="fun6()" id="but06">静音</button>  
<button onclick="fun8()" id="but08">下一首</button>  
<button onclick="fun9()" id="but09">上一首</button>  
  
<!--运动中的进度条-->  
<progress max="100" value="0" style="width: 300px" id="pro\_id" onclick="fun7()"></progress>  
<div id="show"></div>  
</body>

31、this对象  
 指当前事件的目标元素  
 （1）如果在函数中使用this，则this代表什么是根据运行时此函数在什么对象上被调用  
 （即函数中的this是在运行时决定的，而不是在函数定义是决定的）  
 （2）使用apply\call全局方法可以改变函数中this的具体指向

//1  
function fun1(e){  
 //this.style.background='red';//this->window  
 e.style.background='red';  
}  
<button onclick="this.style.background='red'">测试按钮</button>  
<button onclick="fun1(this)">测试按钮</button>  
  
//2  
function fun2(e){  
 event.srcElement.style.background='red'  
}  
<button onclick="event.srcElement.style.background='red'">测试按钮</button>  
<button onclick="fun2(event)">测试按钮</button>

//3  
var v1=12;//全局变量  
function fun3(){  
 var v1=34;  
 alert(this.v1);//12  
}  
fun3();//window.fun3()  
  
var obj={  
 v1:56,  
 method:fun3  
};  
obj.method();//56  
  
//4  
var obj2={  
 v2:23,  
 getv2:function(){  
 alert(this.v2);//23  
 }  
};  
obj2.getv2();  
  
//5  
var v1=12;//全局变量  
function fun3(){  
 var v1=34;  
 alert(this.v1);//12  
}  
fun3();//window.fun3()  
  
var obj={  
 v1:56,  
 method:fun3  
};  
obj.method();//56  
  
obj.method();//56  
fun3.apply(window);//12  
fun3();//12  
fun3.apply(obj);//56  
  
//6  
window.val=1;  
var json={  
 val:10,  
 dbl:function(){  
 this.val\*=2;  
 }  
};  
json.dbl();  
alert(json.val);//20  
  
var dbl=json.dbl;  
dbl();//window.db1()  
alert(window.val);//2  
  
json.dbl.call(window);  
alert(window.val);//4  
  
//7  
window.color='red';  
var obj={  
 color:"yellow"  
};  
function method1(){  
 alert(this.color);  
}  
method1();//red  
obj.m2=method1;  
obj.m2();//yellow  
  
//8  
function fun3(){  
 if(this===window){  
 alert("this 指向 window");  
  
 }  
}  
fun3.m1=function(){  
 if(this===fun3){  
 alert("this 指向 fun3");  
 }else if(this===window){  
 alert("this 指向 window1111");  
 }  
};  
fun3();//this 指向 window  
fun3.m1();//this 指向 fun3  
fun3.m1.apply(window);//this 指向 window1111

32、html5 拖放  
 产生事件的元素 作用  
dragstart: 被拖放的元素 表示开始拖放  
  
dragenter: 被拖放的元素 被拖放元素开始进入区域（开始移动）  
dragover: 被拖放的元素、目标元素 拖放过程中鼠标经过的元素  
dragleave: 被拖放的元素、目标元素 拖放离开时鼠标经过的元素  
  
drop: 目标元素 被拖放元素放下  
dragend: 被拖放的元素 拖放结束  
  
触发顺序：dragstart->dragenter->dragover->dragleave->drop->dragend  
  
被拖放元素需设置 draggable="true"  
  
dataTransfer.setData(key,value) 存储数据  
dataTransfer.getData(key) 提取数据

function fun1(event){  
 //alert("开始拖放");  
 //event.target 事件源  
 //alert(event.srcElement.id);  
 event.dataTransfer.setData("tf\_id",event.srcElement.id)  
  
}  
  
function fun2(){  
 alert("eeeeee");  
}  
  
function fun3(event){  
 alert("eeeeee");  
 event.preventDefault();//阻止默认事件  
}  
function fun4(){  
 alert("eee1111");  
}  
  
function fun5(event){  
 //document.getElementById("id01").appendChild(document.getElementById("id02"));  
 event.preventDefault();//阻止默认事件  
 var v\_tfid=event.dataTransfer.getData("tf\_id");  
 event.srcElement.appendChild(document.getElementById(v\_tfid));  
}

<div id="id02" draggable="true" ondragstart="fun1(event)" ondragenter="fun2()" ondragover="fun3(event)" style="width: 100px;height: 100px;background: blue;"></div> <!--别拖放元素-->  
<br><br>  
<div id="id01" ondragover="fun4()" ondrop="fun5(event)" style="width: 400px;height: 400px;border:1px solid red;"></div> <!--目标元素-->

33、html5 文件  
一、文件选择  
1、多选  
<input type="file" multiple="true">  
2、利用Blob接口获取文件的类型和大小等信息  
size:文件大小  
type:文件的MIME类型，如不能确定字符串大小，返回空字符串  
二、fileReader接口读取文件  
1、该接口可读取图片文件、文本文件、二进制文件，还可以查看文件读取时的详细状态  
2、该接口提供了一个异步API，该API可从浏览器的主线程中异步访问文件系统中的数据，并将数据存入内存  
3、常用方法：  
 readAsBinaryString(f) ：以二进制方式读取文件内容，即将文件以二进制形式写入内存  
 readAsDataURL(f) ：将file对象 返回的数据字节数，以一串数字URL字符的形式展示在“页面里”  
 readAsArrayBuffer(f) ：将file对象 返回的数据字节数，以数组缓冲方式读入内存  
 readAsText(f,encoding) ：以指定的编码方式encoding将读取的数据块按文本方式读入内存，encoding默认utf-8  
  
 abort()：读取文件出现异常，触发该方法

//1  
function fun1(fs){  
 for(var i=0;i<fs.length;i++){  
 alert(fs[i].name+" "+fs[i].size+" "+fs[i].type);  
 //文件小于2M，类型为图片  
 //image/jpg  
 //fs[i].type.match(reg) 或 fs[i].type.exec(reg)  
 }  
}  
  
  
function fun2(fs){  
 if(typeof FileReader=="undefined"){  
 alert("当前浏览器不支持FileReader对象");  
 }  
 //显示照片  
 for(var i=0;i<fs.length;i++){  
 //创建FileReader对象  
 var v\_freader=new FileReader();  
 v\_freader.readAsDataURL(fs[i]);  
 //读取成功时执行onload方法  
 v\_freader.onload=function(event){  
 var esrc=event.srcElement||event.target;  
 document.getElementById("show").innerHTML="<img src='"+esrc.result+"'>";  
 }  
 }  
  
 //显示文件  
 for(var i=0;i<fs.length;i++){  
 //创建FileReader对象  
 var v\_freader=new FileReader();  
 v\_freader.readAsText(fs[i],"utf-8");  
 v\_freader.onload=function(event){  
 var esrc=event.srcElement||event.target;  
 document.getElementById("show").innerHTML=esrc.result;  
 }  
 }  
}  
  
//2  
function fun3(event){  
 event.preventDefault();  
 event.stopPropagation();  
 var fs=event.dataTransfer.files;  
 var s="";  
 var t="";  
 for(var i=0;i<fs.length;i++){  
 s+=(i+1)+"、"+fs[i].name+" "+fs[i].size+" "+fs[i].type+"<br>";  
 var reg=/image/gi;  
 if(fs[i].type.match(reg)){  
 var v\_freader=new FileReader();  
 v\_freader.readAsDataURL(fs[i]);  
 v\_freader.onload=function(event){  
 var esrc=event.srcElement||event.target;  
 var img=document.createElement("img");  
 var br=document.createElement("br");  
 img.setAttribute("src",esrc.result);  
 document.getElementById("show").appendChild(img);  
 document.getElementById("show").appendChild(br);  
 }  
 }  
 }  
 event.srcElement.innerHTML=s;  
}

<!--1-->  
<fieldset>  
 <legend>上传单个文件</legend>  
 <input type="file">  
  
 <legend>上传多个文件</legend>  
 <input type="file" multiple="true" onchange="fun1(this.files)">  
  
 <legend>上传多个文件并显示</legend>  
 <input type="file" multiple="true" onchange="fun2(this.files)">  
</fieldset>  
<div id="show"></div>  
  
<!--2-->  
<fieldset>  
 <legend>拖放文件</legend>  
 <div id="id01" ondrop="fun3(event)" ondragover="return false;" ondragenter="return false;" style="width: 500px;height: 100px;border:1px solid red"></div>  
</fieldset>  
<div id="show"></div>

34、html5 离线应用  
1、离线应用文件就是一个URL列表，在该列表中包括：html、css、js、图片及其他资源型文件  
2、该文件扩展名 \*.manifest  
3、当应用程序与服务器建立联系时，浏览器将在本地缓存所有URL列表（离线应用配置文件）中的资源文件  
 当应用程序与服务器失去联系时（离线状态），浏览器将调用本地缓存的文件来继续支撑web应用  
补充3：html5提供一种缓存机制---Cache Manifest，可在线时将对应文件缓存到本地  
 离线时调用这些本地文件  
离线配置文件说明：  
CACHE MANIFEST：表示这是一个通过浏览器将服务器资源进行本地缓存的格式文件  
  
CACHE：该标记表示离线时，浏览器需要缓存到本地的服务器资源文件列表  
  
NETWORK：该标记表示在线时，需要访问的资源文件列表，这些文件只有浏览器与服务器  
 建立联系时才能访问  
  
FALLBACK:该标记表示以成对方式列出不可访问文件的替补文件 "aa.html" "bb.html"  
  
manifest="testline.manifest" 通过页面的<html>元素的manifest属性，将页面与testline.manifest文件进行  
 关联，在浏览器查看页面时，自动将testline.manifest文件中涉及的资源文件  
 缓存到本地。  
  
状态更新方法：  
 applicationCache  
 检测本地缓存是否完成更新  
 updateready()  
 update()手动更新

<link href="../css/test30.css" rel="stylesheet">  
<script src="../js/test30.js"></script>

function fun\_load(){  
 alert(window.applicationCache.state);  
 window.applicationCache.addEventListener("updateready",function(){  
 alert("正在触发更新事件。。。。")  
 },false);  
  
 /\*  
 \* 监测网络的当前状态  
 \* online 在线  
 \* offline 离线  
 \* \*/  
 window.addEventListener("online",function(){  
 alert("当前网络在线。。。。")  
 },false);  
 window.addEventListener("offline",function(){  
 alert("当前网络离线。。。。")  
 },false);  
}  
function fun3(){  
 window.applicationCache.update();  
}  
  
/\*  
\* 离线留言板  
\* online或offline检测当前网络  
\* 如是离线状态，先将留言的数据暂时存储在本地localstroage对象中  
\* 上线时，再将存储的数据传智服务器  
\* \*/

<body onload="fun\_load()">  
<div class="div01">hello outline application!</div>  
当前时间：<div id="curtime"></div>  
<button onclick="fun3()">手动更新离线缓存</button>  
</body>

testline.manifest内容：

CACHE MANIFEST  
#version 0.0.0  
  
CACHE:  
../css/test30.css  
../js/test30.js

test30.js内容：

function fun1(){  
 var v1=new Date();  
 var v2=v1.getHours();  
 var v3=v1.getMinutes();  
 var v4=v1.getSeconds();  
 if(v2<10){  
 v2="0"+v2;  
 }  
 if(v3<10){  
 v3="0"+v3;  
 }  
 if(v4<10){  
 v4="0"+v4;  
 }  
 document.getElementById("curtime").innerHTML=v2+":"+v3+":"+v4;  
}  
setInterval(fun1,1000);

test30.css内容：

.div01{  
 width: 400px;  
 height: 400px;  
 border:1px solid #000000;  
 font-size: 30px;  
}  
#curtime{  
 width: 100px;  
 height: 30px;  
 line-height: 30px;  
 text-align: center;  
 background: chartreuse;  
}

50、html5 websocket //socket 推送  
\* websocket：1、实现了浏览器与服务器“全双工”通信  
\* 2、是html5的新协议  
\* 3、服务器可以主动传送数据给客户端（推送）

var ws;  
function fun\_init(){  
 if(window.WebSocket){  
 alert("当前浏览器支持WebSocket");  
 }else{  
 alert("当前浏览器不支持WebSocket");  
 }  
 //创建websocket  
 //WebSocket 或 MozWebSocket  
 if('WebSocket' in window){  
 ws=new WebSocket("ws://127.0.0.1:8888/serverphp.php");  
 }else if("MozWebSocket" in window){  
 ws=new MozWebSocket("ws://127.0.0.1:8888/serverphp.php");  
 }  
 //接收服务器端数据  
 ws.onmessage=function(msg){  
 alert(msg);  
 }  
}  
function fun\_send(){  
 var v1=document.getElementById("inp01").value;  
 //利用WebSocket发送数据  
 try{  
 ws.send(v1);  
 }catch(e){  
 alert(e);  
 }  
}

<body onload="fun\_init()">  
<input type="text" id="inp01">  
<button onclick="fun\_send()">发送</button>  
</body>

test31\_socket.php内容：

<?php  
$v\_ip="127.0.0.1";  
$v\_port=8888;  
  
//生成一个socket  
$v\_socket=socket\_create(AF\_INET,SOCK\_STREAM,SOL\_TCP);  
if($v\_socket<0){  
 echo "socket\_create创建失败，错误：".socket\_strerror($v\_socket);  
}  
//绑定ip与端口  
$v\_bind=socket\_bind($v\_socket,$v\_ip,$v\_port);  
if($v\_bind<0){  
 echo "socket\_bind创建失败，错误：".socket\_strerror($v\_bind);  
}  
//监听  
$v\_listen=socket\_listen($v\_socket,4);  
if($v\_listen<0){  
 echo "socket\_listen创建失败，错误：".socket\_strerror($v\_listen);  
}  
echo "listening on.....".$v\_ip." ".$v\_port;  
do{  
 //获取客户端连接  
 $v\_client=socket\_accept($v\_socket);  
 if($v\_client<0){  
 echo "socket\_accept失败！";  
 }else{  
 echo "\n...client...";  
 //向客户端发送数据  
 $msg="hello client!";  
 socket\_write($v\_client,$msg,strlen($msg));  
  
 //接收客户端数据  
 $str=socket\_read($v\_client,8192);  
 echo $str;  
 }  
 socket\_close($v\_client);  
}while(true);  
  
socket\_close($v\_socket);  
?>

51、 CSS3 选择器  
E~F ：匹配任何E元素的同级元素F  
E:nth-child() ：选择所有其父元素中第n个位置的匹配E元素  
 索引值由1开始  
 参数：正常数字1/2/3/4  
 odd even 2n 2n+1 3n 3n+2 ....  
 (n由0开始)  
E:nth-of-type() ：选择父元素中第n个位置，且匹配E的子元素---将所有匹配的子元素分离出来单独排序  
  
E:nth-last-child() ：匹配父元素的倒数第n个子元素E，假设该子元素不是E，则选择符无效。  
E:nth-last-of-type() ：匹配同类型中的倒数第n个同级兄弟元素E。  
  
E:last-child ：匹配父元素的最后一个子元素E。  
E:last-of-type ：匹配同类型中的最后一个同级兄弟元素E。  
  
E:first-child ：匹配父元素的第一个子元素E。  
E:first-of-type ：匹配同类型中的第一个同级兄弟元素E。  
  
E:only-child ：匹配父元素仅有的一个子元素E。  
E:only-of-type ：匹配同类型中的唯一的一个同级兄弟元素E。  
  
E:empty ：匹配E元素，该元素不包含任何子节点，包括文本节点（文本也属于子节点）  
  
E:not(s) ：匹配不含s选择符的元素E  
  
//css3 新增伪类  
E:target ：匹配文档中特定的“id”点击效果（超链接锚点例子）  
  
//属性选择器  
E[att^=val]：属性att以val开头的元素  
E[att$=val]：属性att以val结尾的元素  
E[att\*=val]：属性att包含val的元素

/\*1\*/  
div{  
 color:red;  
}  
.d1~div{  
 color:#ffff00;;  
}  
  
.d2 div:nth-child(2n){  
 color:#ffff00;  
 background: red;  
 }  
  
.d3 span:only-child{  
 color:#ffff00;  
 background: red;  
}  
  
div:empty{  
 width: 100px;  
 height: 100px;  
 background: red;  
}  
  
ul li:not(:last-child){  
 color: red;  
}  
  
#id01,#id02,#id03{  
 width: 300px;  
 height: 200px;  
}  
#id01{  
 background: red;  
}  
#id02{  
 background: blue;  
}  
#id03{  
 background: yellow;  
}  
#ida{  
 position: fixed;  
 left: 500px;  
}  
div:target{  
 border:3px solid #000000;  
}  
  
/\*2\*/  
div[id\*='id']{  
 color:red;  
}

<!--1-->  
<div>kkkk</div>  
<div class="d1"></div>  
<div class="d2">  
 bbbb  
 <div>mmmm</div>  
 <div>mmmm</div>  
 <span>ssss</span>  
 <div>mmmm</div>  
 <div>mmmm</div>  
 <div>mmmm</div>  
 <div>mmmm</div>  
 <div>mmmm</div>  
 <div>mmmm</div>  
 <div>mmmm</div>  
</div>  
<div class="d3">  
 <span>ssss</span>  
</div>  
<div></div>  
  
<ul>  
 <li>aaaaaaa</li>  
 <li>bbbbbbb</li>  
 <li>ccccccc</li>  
 <li>ddddddd</li>  
</ul>  
  
<div id="ida">  
 <!--锚点-->  
 <a href="#id01">链接1</a>  
 <a href="#id02">链接2</a>  
 <a href="#id03">链接3</a>  
</div>  
<div id="id01"></div>  
<div id="id02"></div>  
<div id="id03"></div>  
  
<!--2-->  
<div id="id04" title="test11111111">test11111.....</div>  
<div id="aid05" title="test11111111">test22222.....</div>  
<div id="06id" title="test11111111">test33333.....</div>

52、CSS3：新增色彩模式  
RGBA()  
 A-opacity 0-1  
 opacity透明度 0-1  
  
HSL(,,)：H-色调 -360 -0 -360  
 0 红色 60 黄色 120 绿色 180 青色 240 蓝色  
 S-饱和度 百分比 0%-表示灰度，即没有使用该颜色  
 100%-饱和度最高，颜色最鲜艳  
 L-亮度 百分比 0%-最暗，黑色  
 50%-平均色  
 100%-最亮，白色  
  
HSLA() A-透明度

.id01{  
 width: 200px;  
 height: 40px;  
 background: rgba(255,0,0,0.5);  
 /\*高版本\*/  
 opacity: 0.5;  
 /\*低版本\*/  
 filter:alpha(opacity=50);  
  
 background-color: HSL(0,100%,50%);  
  
 background-color: HSLA(120,50%,50%,0.1);  
}  
.id02{  
 width: 200px;  
 height: 40px;  
 background-color: HSLA(120,50%,50%,0.2);  
}  
.id03{  
 width: 200px;  
 height: 40px;  
 background-color: HSLA(120,50%,50%,0.3);  
}

<div class="id01">hello</div>  
<div class="id02">hello</div>  
<div class="id03">hello</div>

53、css3 渐变

.d1{  
 width: 300px;  
 height: 300px;  
 border:2px solid red;  
 /\*线性渐变 -webkit-  
 background: -webkit-gradient(linear,起始点,终止点,起始颜色,终止颜色,中间过渡色1,中间过渡色2.。。。。);  
 \*/  
 background: -webkit-gradient(linear,20% 40%,40% 50%,from(red),to(blue),color-stop(0.3,#ffff00),color-stop(0.6, #28ff1f));  
  
 /\*线性渐变 -moz- GECKO\*/  
 background: -moz-linear-gradient(red,blue);  
  
 /\*径向渐变 -webkit-  
 background: -webkit-gradient(radial,内圆坐标,内圆半径,外圆坐标,外圆半径,起始颜色,终止颜色,中间过渡色。。。。);  
 \*/  
 background: -webkit-gradient(radial,150 150,100,150 150,150,from(red),to(blue),color-stop(0.3,gold));  
  
 /\*径向渐变 -moz- GECKO  
 background: -webkit-gradient(radial,内圆坐标,内圆半径,外圆坐标,外圆半径,起始颜色,终止颜色,中间过渡色。。。。);  
 \*/  
 background: -moz-radial-gradient(red,blue);  
}

<div class="d1"></div>

35、css3 文本字体

/\*使用外挂字体\*/  
@font-face {  
 font-family: myfont;  
 src:url("DK TARTUFO.OTF")format("OpenType");  
}  
.d1{  
 font-size: 30px;  
 font-family: "黑体";  
 /\*字体阴影  
 text-shadow:相对于原字体X方向偏移量，相对于原字体Y方向偏移量，模糊度，阴影颜色  
 \*/  
 text-shadow: 2px 3px 4px red;  
 text-shadow: 0 0 4px white,  
 0 -5px 4px yellow,  
 2px -10px 6px #e3e300,  
 -2px -15px 11px #ff281c,  
 2px -25px 18px #e70217;  
 text-shadow: -1px -1px white,  
 1px 1px #333;  
  
 /\*字体边框\*/  
 text-stroke:2px red;  
 -webkit-text-stroke:2px red;//webkit  
 -moz-text-stroke:2px red;//firefox  
 -ms-text-stroke:2px red;//ie  
 -o-text-stroke:2px red;//opera  
  
 /\*文字填充\*/  
 -webkit-text-fill-color:transparent;//填充透明  
  
 /\*渐变背景文字\*/  
 width: 300px;  
 height: 300px;  
 background: -webkit-gradient(linear,left top,left bottom,from(red),to(blue),color-stop(0.3,yellow));  
 -webkit-background-clip: text;  
 -webkit-text-fill-color: transparent;  
  
 /\*  
 word-break:让文本自动换行  
 break-all:允许在单词内换行  
 keep-all:只能在半角空格或练字符除换行  
 normal:默认，使用浏览器默认的换行规则  
  
 word-wrap:让长单词或文本超出指定边界时断开换行  
 break-word  
  
 text-overflow:文本溢出处理方式  
 前提：内容不折行  
 ellipsis：显示省略标记（...）  
 clip:不显示省略标记，直接裁切  
 \*/  
 width: 300px;  
 height: 300px;  
 border: 1px solid red;  
 word-wrap:break-word;  
 word-break:break-all;  
  
 overflow: hidden;  
 text-overflow:ellipsis;  
  
 /\*使用外挂字体\*/  
 font-family: myfont;  
  
 /\*  
 font-size-adjust:设置或检索对象的aspect值  
 以字体里x的高度与字号之间的比率--aspect值 x--height  
 原因：如果改变了字体种类，则页面中所有使用该字体的字号大小都可能发生变化，使原布局混乱  
 \*/  
 font-size: 16px;  
 font-family: 'Times New Roman';  
 font-family:"Comic Sans MS";/\*aspect值 0.54\*/  
 font-size-adjust: 0.49;/\*'Times New Roman'的aspect值\*/  
}

<div class="d1">dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd</div>  
<div class="d1">hello 你好 hello 你好 hello 你好 hello 你好 hello 你好 hello 你好 hello 你好 hello 你好 hello 你好 hello 你好 hello 你好 hello 你好 hello 你好 </div>  
<div class="d1">hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello</div>

36、css3边框与背景

.d1{  
 width: 300px;  
 height: 300px;  
 border: 2px solid red;  
 background: #034a76;  
 /\*盒子阴影  
 box-shadow:阴影类型 X方向偏移量 Y方向偏移量 阴影模糊度 阴影大小 阴影颜色;  
 inset 内阴影  
 \*/  
 box-shadow: 2px 3px 10px 10px blue;  
 box-shadow:inset 2px 3px 10px 10px blue;  
  
 box-shadow: 0 0 1px #fff,  
 0 0 3px #000,  
 0 0 3px #000,  
 inset 0 1px #fff,  
 inset 0 12px rgba(255,255,255,0.1),  
 inset 0 4px 10px #cef,  
 inset 0 22px 5px #0773b4,  
 inset 0 -5px 10px #0df;  
}  
  
.d2{  
 /\*  
 background-clip:指定背景的显示范围  
 text:显示文本范围的背景  
 border:背景范围包括边框区域  
 padding:背景范围不包括边框区域  
  
 background-origin:指定背景图像的绘制起点  
 border-box:左上角  
 padding-box:内部补白区域的左上角  
 content-box:内容的左上角  
  
 background-size:指定背景图像尺寸  
 100px 100px  
 50% 50%  
 100% 100%  
 auto 50%  
 cover:背景图片等比缩放到完全覆盖容器，可能会超出容器  
 contain:背景图片等比缩放与容器的宽高相等，但不会超出容器  
 \*/  
 width: 200px;  
 height: 200px;  
 border:30px solid transparent;  
   
 border:3px solid red;  
 padding: 50px;  
 background: url(../images/Dior01.jpg);  
  
 -webkit-background-clip: padding;  
 -webkit-background-clip: border;  
  
 -webkit-background-origin: border-box;  
 -webkit-background-origin: padding-box;  
 -webkit-background-origin: content-box;  
  
 background-size: 100% 100%;  
  
 /\*多重背景\*/  
 background: url(../images/Dior01.jpg) no-repeat,  
 url(../images/Dior03.jpg) no-repeat;  
}  
  
.d3{  
 /\*边框  
 border-radius  
 \*/  
 width: 200px;  
 height: 200px;  
 border:1px solid red;  
 border-radius: 10px; //四边  
 border-radius: 10px 20px; //上左 下右  
 border-radius: 10px 20px 30px 40px; //顺时针 上左 上右 下右 下左  
 border-radius: 20px 0 0 0/40px 0 0 0; //水平半径 垂直半径  
}

<div class="d1"></div>  
<div class="d2">hello</div>  
<div class="d3"></div>

37、

/\*:before :after\*/  
.d1{  
 width: 300px;  
 height: 70px;  
 border:1px solid red;  
}  
.d1:before{  
 content: "aaaaaaa";  
 color:red;  
 font-size: 30px;  
}  
.d1:after{  
 content: "aaaaaaa";  
 color:red;  
 font-size: 30px;  
}  
  
ul li:nth-child(1):before{  
 content: url(../images/Dior01.jpg);  
}  
  
/\*计数器\*/  
/\*定义计数器\*/  
p{  
 counter-increment: mycounter;  
}  
p:before{  
 content: counter(mycounter)"、";  
 color: red;  
}

<div class="d1">hello</div>  
  
<ul>  
 <li>aaaaaaaaa111</li>  
 <li>aaaaaaaaa222</li>  
 <li>aaaaaaaaa333</li>  
</ul>

<p>aaaaaaaaaaaaaa</p>  
<div>ddddddddddd</div>  
<p>bbbbbbbbbbbbbb</p>  
<p>cccccccccccccc</p>

38、

.box{  
 width: 300px;  
 height: 300px;  
 background: #fff;  
 border:1px solid #ccc;  
 position: relative;  
 top:100px;  
 left: 100px;  
 background: -webkit-gradient(linear,0% 20%,0%

100%,from(#fff),to(#fff),color-stop(0.1,#f3f3f3));  
 box-shadow:inset 0 3px 30px rgba(0,0,0,0.1);  
  
 border-radius: 0 0 6px 0/0 0 50px 0;  
 /\*或：\*/  
 border-top-left-radius:  
 border-bottom-rght-radius:  
  
}  
.box:before{  
 content: '';  
 width: 50px;  
 height: 100px;  
 box-shadow: 20px 20px 15px rgba(0,0,0,0.1);  
 position: absolute;  
 bottom: 0;  
 right: 0;  
 transform: translate(-35px,-40px) skew(0deg,32deg) rotate(-25deg);  
 z-index: -1;  
}  
.box:after{  
 content: '';  
 width: 100px;  
 height: 100px;  
 position: absolute;  
 top: 0;  
 left: 0;  
 box-shadow: -10px -10px 15px rgba(0,0,0,0.2);  
 transform: translate(20px,25px) skew(20deg) rotate(7deg);  
 z-index: -1;  
}  
  
.box{  
 width: 100px;  
 height: 100px;  
 border-width:40px;  
 border-style: solid;  
  
 border-color: red;  
 border-left-color: green;  
 border-right-color: blue;  
 border-top-color: #ffff00;  
 border-bottom-color: palevioletred;  
  
 /\*border-colors 仅支持火狐  
 -moz-border-left-colors:  
 -moz-border-right-colors:  
 ....  
 当属性只有一个颜色值时，表示纯色  
 当有n个颜色值时，每个颜色显示1px宽，最后一种颜色将显示剩余的边框宽度  
 \*/  
 -moz-border-left-colors:red #33cd6f #ffff00 #245 #067;  
  
 /\*border-image\*/  
 border-image: url("../images/Dior01.jpg") 20 repeat;  
 border-image: url("../images/Dior01.jpg") 20 40 repeat;  
 border-image: url("../images/Dior01.jpg") 20 40 60 80 repeat;  
}

<div class="box"></div>

39、开关

.slidekg{  
 width: 150px;  
 height: 30px;  
 position: relative;  
 border-radius: 50px;  
 box-shadow: inset 0 1px 1px rgba(0,0,0,0.5),  
 0 1px 0 rgba(255,255,255,0.2);  
 /\*border:1px solid rgba(255,255,255,0.3);\*/  
}  
.slidekg label{  
 width: 26px;  
 height: 26px;  
 border-radius:26px;  
 position: absolute;  
 top:2px;  
 left: 2px;  
 /\*background: -webkit-linear-gradient(top,#feffe1 0%,#dfef71 40%);\*/  
 background: -webkit-gradient(linear,0% 20%,0% 100%,from(red),to(blue),color-stop(0.3,#ffff00),color-stop(0.1, #f3f3f3));  
 box-shadow:0 2px 5px 0 rgba(0,0,0,0.3);  
 -webkit-transition: all 0.4s ease;  
 cursor: pointer;  
}  
.slidekg input[type="checkbox"]{visibility: hidden;}  
  
.slidekg input[type="checkbox"]:checked+label{  
 left: 120px;  
}

<div class="slidekg">  
 <input type="checkbox" id="chek01" checked>  
 <label for="chek01"></label>  
</div>

40、变形transform

.d1{  
 width: 100px;  
 height: 100px;  
 background: red;  
 background: url("../images/Dior01.jpg");  
}  
.d1:hover{  
 /\*缩放 scale  
 scale(水平方向缩放,垂直方向缩放)  
 参数可以是正数、小数、负数  
 如参数为负数，不仅缩放元素还翻转元素  
 \*/  
 transform:scale(0.5);  
 /\*  
 scale(0.5,2)  
 scaleX(0.5)  
 scaleY(5)  
  
 scale(-1) 翻转180度  
 scale(-1,-1) 翻转180度  
 scale(-1,1) 水平翻转  
 scale(1,-1) 垂直翻转  
 \*/  
  
 /\*倾斜 skew  
 skew(X轴水平缩放,Y轴方向垂直缩放)  
 \*/  
 transform: skew(30deg);  
 /\*  
 skew(-30deg)  
 skew(30deg,70deg)  
  
 skewX(30deg)  
 skewY(30deg)  
 \*/  
  
 /\*旋转 rotate\*/  
 transform: rotate(45deg);  
 /\*  
 rotate(-45deg)  
 rotateX(45deg)  
 rotateY(45deg)  
 rotateZ(45deg)  
 \*/  
  
 /\*移动 translate\*/  
 transform: translate(40px);  
 /\*  
 transform: translate(40px,90px);  
 transform: translate(-40px,-20px);  
 transform: translateX(40px);  
 transform: translateY(40px);  
 \*/  
  
 /\*整合应用\*/  
 transform: translate(40px,50px) scale(2) rotate(45deg) skew(60deg);  
  
 /\*设置变形基准点\*/  
 transform: rotate(45deg);  
 transform-origin: left bottom;  
 /\*  
 水平：left center right  
 垂直：top center bottom  
 10px 20px  
 100% 50px  
 \*/  
}

<div class="d1"></div>

42、过渡 transition  
transition: 要过渡的属性（all 所有属性） 过渡时间 过渡方式 过渡延迟;  
transition: background 3s linear 3s;  
 linear 线性过渡，从头到尾速度相同  
 ease 平滑过渡  
 ease-in 由慢变快  
 ease-out 由块到慢  
 ease-in-out 由慢到块再到慢

.d1{  
 width: 300px;  
 height: 300px;  
 border: 5px solid red;  
 background: red;  
 background: url(../images/aa.jpg);  
 transition: background 3s ease 3s,  
 width 3s linear;  
 /\*-webkit-transition: ;\*/  
 /\*-moz-transition: ;\*/  
 /\*-ms-transition: ;\*/  
 /\*-o-transition: ;\*/  
}  
.d1:hover{  
 background: blue;  
 width: 400px;  
   
 border: 10px solid blue;  
 transform: rotateY(360deg);  
 transition: all 3s linear;  
}

<div class="d1"></div>

43、animations 动画  
原理：通过定义多个关键帧以及定义每个关键帧中元素的属性值，来实现复杂的动画效果

.d1{  
 width:300px;  
 height: 300px;  
 background: red;  
}  
/\*定义帧\*/  
@-webkit-keyframes myani{  
 0%{background: red;width: 400px;}  
 40%{background: blue;width: 200px;}  
 70%{background: #ffff00;width: 300px;}  
 100%{background: green;width: 400px;}  
  
 from{transfrom:rotate(0deg)}  
 to{transfrom:rotate(0deg)}  
}  
.d1:hover{  
 /\*设置动画\*/  
 -webkit-animation-name: myani;/\*动画名称\*/  
 -webkit-animation-duration: 5s;/\*动画持续时间\*/  
 -webkit-animation-timing-function: linear;/\*动画速度曲线（动画过滤类型）  
 linear 动画 从头到尾速度相同  
 ease 低速开始，后加快，在结束前变慢  
 ease-in 动画以低速开始  
 ease-out 动画以低速结束  
 ease-in-out 动画以低速开始和结束  
 \*/  
 -webkit-animation-iteration-count:infinite;/\*循环次数 infinite无线循环\*/  
 -webkit-animation-delay: 2s;/\*动画开始前的延迟\*/  
 -webkit-animation-direction: alternate;/\*是否轮流反向播放

normal:默认正常播放 alternate:反向播放\*/  
 /\*简写\*/  
 animation: myani 5s ease infinite;  
 -webkit-animation: myani 5s ease infinite;  
 -moz-animation: myani 5s ease infinite;  
 -o-animation: myani 5s ease infinite;  
 -ms-animation: myani 5s ease infinite;  
}

<div class="d1"></div>

/\*半圆\*/  
.d2{  
 width: 100px;  
 height: 100px;  
 border-width: 15px 15px 0 0;  
 border-style: solid;  
 border-color: red;  
 transform: rotate(-45deg) translate(-100px,150px);  
 border-radius: 0 100% 0 ;  
}

<div class="d2"></div>

44、css3 3D  
perspective: 2000px; 透视/景深，仅作用于该元素的后代  
transform-style: preserve-3d; 开启3d效果，仅作用于该元素的后代

@-webkit-keyframes myani{  
 0%{transform: rotateX(0deg);}  
 25%{transform: rotateY(-90deg);}  
 50%{transform: rotateY(40deg);}  
 75%{transform: rotateX(90deg);}  
 100%{transform: rotateX(120deg);}  
}  
.wrap{  
 width: 500px;  
 height: 500px;  
 position: relative;  
 left: 50%;  
 top: 50%;  
 perspective: 2000px;  
}  
.wrap .box{  
 transform-style: preserve-3d;  
 -webkit-animation: myani 5s linear infinite;  
}  
.box div{  
 width: 100px;  
 height: 100px;  
 border:1px solid red;  
 position: absolute;  
 top: 300px;  
 background: rgba(255,0,0,0.3);  
}  
.left{  
 transform: rotateY(-90deg) translateZ(50px);  
}  
.right{  
 transform: rotateY(90deg) translateZ(50px);  
}  
.top{  
 transform: rotateX(90deg) translateZ(50px);  
}  
.bottom{  
 transform: rotateX(-90deg) translateZ(50px);  
}  
.front{  
 transform:translateZ(50px);  
}  
.back{  
 transform:translateZ(50px);  
}

<div class="wrap">  
 <div class="box">  
 <div class="front">1</div>  
 <div class="back">2</div>  
 <div class="left">3</div>  
 <div class="right">4</div>  
 <div class="top">5</div>  
 <div class="bottom">6</div>  
 </div>  
</div>

45、css3 多列

.d0{  
 -webkit-column-count: 4;  
 -webkit-column-gap: 40px;/\*列间距\*/  
 -webkit-column-width: 200px;/\*列宽\*/  
  
 /\*列边框\*/  
 -webkit-column-rule-width: 10px;  
 -webkit-column-rule-color: red;  
 -webkit-column-rule-style: dotted;  
 /\*简写\*/  
 -webkit-column-rule:10px dotted red;  
}  
.d1{  
 font-size: 40px;  
 /\*横跨所有列\*/  
 -webkit-column-span: all;  
 text-align: center;  
}

<div class="d0">  
 <div class="d1">...XXX小说...</div>  
 奉天老王府的.....人已经在城门口等他们了</div>

46、弹性盒子模型 box

.d1,.d2,.d3{  
 border:1px solid red;  
}  
.d0{  
 display: box;/\*开启弹性盒子模型\*/  
 display: -webkit-box;  
  
 box-orient: horizontal;/\*盒子布局取向 horizontal 元素从左到右水平排列  
 vertical 元素从上到下纵向排列\*/  
 -webkit-box-orient: horizontal;  
  
 -webkit-box-direction: reverse;/\*设置元素是否反向布局 reverse 反向\*/  
  
 /\*空间管理\*/  
 /\*  
 box-pack 剩余空间分配--水平  
 center:剩余空间在盒子两侧均匀分配  
 start:设置子元素从开始对齐  
 end:设置子元素从结束对齐  
 justify:剩余空间在子元素之间均匀分配，但第一个子元素之前和最后一个子元素之后不分配空间  
 \*/  
 -webkit-box-pack: justify;  
  
 /\*  
 box-align:剩余空间分配--垂直  
 center:剩余空间在盒子的上下两侧平均分配  
 stretch:每个元素的高度被调整到适合盒子的高度显示  
 start:所有子元素沿盒子的上边缘排列，剩余空间显示在盒子底部  
 end:所有子元素沿盒子的下边缘排列，剩余空间显示在盒子顶部  
 \*/  
 -webkit-box-align: stretch;  
  
 /\*  
 box-lines:子元素是否可以换行显示  
 single:一行显示  
 multiple:超出父元素空间换行显示  
 \*/  
 box-lines:multiple;  
 -webkit-box-lines:multiple;  
 -o-box-lines:multiple;  
 -moz-box-lines:multiple;  
}  
img{  
 height: 100px;  
}

<div class="d0">  
 <!--<div class="d1"><img src="../images/Dior02.jpg"></div>-->  
 <!--<div class="d2"><img src="../images/Dior01.jpg"></div>-->  
 <!--<div class="d3"><img src="../images/Dior01.jpg"></div>-->  
  
 <div class="d1">1</div>  
 <div class="d2">2</div>  
 <div class="d3">3</div>  
</div>

47、

<!--页面宽度适应设备宽度-->  
<meta name="viewport" content="initial-scale=1,user-scalable=no,width=device-width">

（1）

/\*响应式设计 media  
响应式设计--responseive web design,识别屏幕尺寸，做出页面的响应调整（自适应设计）  
 max-width:最大宽度，指浏览器小于这个最大宽度时采用的样式  
 min-width:最小宽度，指浏览器大于这个最小宽度时采用的样式  
  
 参数：  
 all:表示所有媒体设备  
 screen:表示计算机显示器  
 tv:电视类设备  
 handheld:小型或手持设备  
\*/  
.d0{  
 width: 400px;  
 height: 400px;  
 border:1px solid red;  
 background: red;  
}  
@media all and (max-width:500px){  
 .d0{  
 background: #007aff;  
 }  
}  
@media all and (min-width:500px) and (max-width:800px){  
 .d0{  
 background: #ffff00;  
 }  
}

（2）

<style media="all and (max-width:500px)">  
 ........  
</style>

（3）

<link href="../css/css1.css" rel="stylesheet" media="all and (max-width:500px)">

.d0{  
 border:1px solid #000000;  
 width: 100%;  
 position: absolute;  
 height: 100%;  
 display: -webkit-box;  
 -webkit-box-orient: horizontal;  
}  
.d1,.d2{  
 border:1px solid red;  
}

<!--<div class="d0"></div>-->  
  
<div class="d0">  
 <div class="d1">1</div>  
 <div class="d2">2</div>  
</div>

48、css3 流布局（flex）  
 display: flex; 开启流布局  
 flex-grow: 设置盒子的扩展比例，默认值0，不分配剩余空间  
 盒子所有flex-grow相加，根据各自的比重分配剩余空间  
 flex-shrink: 盒子的收缩比例  
 flex-basis: 盒子的伸缩基准值，如基准值之和大于或小于父元素，剩余空间会根据基准值的比值进行剩余空间的缩放分配  
  
 简写：flex：flex-grow flex-shrink flex-basis

.d0{  
 width: 900px;  
 height: 500px;  
 border:1px solid #000000;  
 display: flex;/\*开启流布局\*/  
 flex-direction: row;/\* row row-reverse column column-reverse\*/  
  
 /\*align-items 子元素的垂直对齐方式  
 flex-start 上对齐  
 flex-end 下对齐  
 stretch 拉伸  
 \*/  
 align-items: center;  
  
 /\*justify-content 子元素的水平对齐方式  
 flex-start 左对齐  
 flex-end 右对齐  
 space-around 子元素平均分配剩余空间  
 space-between 子元素平均分配剩余空间，但第一个元素之前和最后一个元素之后不分配剩余空间  
 \*/  
 justify-content: center;  
}  
.d1,.d2,.d3{  
 border:1px solid red;  
}  
.d1{  
 flex-grow: 1;  
 flex-shrink: 1;  
 flex-basis: 300px;  
  
 /\*flex:1 1 300px;\*/  
}  
.d2{  
 flex-grow: 1;  
 flex-shrink: 2;  
 flex-basis: 400px;  
}  
.d3{  
 flex-grow: 1;  
 flex-shrink: 3;  
 flex-basis: 500px;  
}  
/\*  
300+400+500=1200  
1200-900=300  
300\*1+400\*2+50\*3=2600  
(300\*1/2600)\*300=35 300-35=265  
(400\*1/2600)\*400=92 400-93=308  
(500\*1/2600)\*500=173 500-173=327  
\*/

<div class="d0">  
 <div class="d1">1</div>  
 <div class="d2">2</div>  
 <div class="d3">3</div>  
</div>