AngularJS--第四天 课程

事件指令

ng-开头的件指令可以支持表达式数据获取

ng-click：点击事件
ng-dblclick：双击事件
ng-mousedown：鼠标按下事件
ng-mouseup：鼠标抬起事件
ng-mouseenter：鼠标穿过事件
ng-mouseleave：鼠标穿出事件
ng-mousemove：鼠标移动事件
ng-keydown：键盘按下事件
ng-keyup：键盘抬起事件
ng-focus：焦点事件
ng-blur：失去焦点事件

ng-selected：选中下拉列表框

案例

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


    }])
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
        <input type="checkbox" ng-model="yes">
        <select>
            <option>请仔细阅读相关文件</option>
            <option ng-selected="yes">我已经阅读过相关文件</option>
        </select>
    </div>
</body>
</html>
```

ng-change:输入内容触发

案例

```
<!DOCTYPE html>
```

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


   }])
   </script>
</head>
<body>
   <div ng-controller='Aaa'>
      <input type="text" ng-change="name='change1'" ng-model="name">
      <p>{{name}}</p>
   </div>
</body>
</html>
```

ng-copy：复制内容触发

ng-cut：剪切内容触发

ng-paste：粘贴内容触发

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <script src="js/angular.min.js"></script>
   <script>
   var m1 = angular.module('myApp',[]);
   m1.controller('Aaa',["$scope",function($scope){
      $scope.name='hello';
   }])
   </script>
</head>
<body>
   <div ng-controller='Aaa'>
      <!-- <input type="text" value="saigonw" ng-cut="cut=true">{{cut}} -->
      <!-- 进行剪切动作后触发条件 -->
      <!-- <input type="text" value="saigonw" ng-copy="copy=true">{{copy}} -->
      <!-- 进行复制动作后触发条件 -->
      <!-- <input type="text" value="saigonw" ng-paste="paste=true">{{paste}} -->
       <!-- 进行粘贴动作后触发条件 -->
   </div>
</body>
</html>
```

ng-disabled：控制（控件）能否使用，让我们可以动态的修改控件状态。

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <script src="js/angular.min.js"></script>
   <script>
```

```
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',["$scope","$interval",function($scope,$interval){
        var Num = 5;
        $scope.name=Num+'秒';
        $scope.isDisabled=true;

        var timer=$interval(function(){
            Num--;
            $scope.name=Num+'秒';

            if(Num == 0){
                $interval.cancel(timer);
                $scope.name='按钮可以使用了';
                $scope.isDisabled=false;
              }
        },1000);

        // setInterval()  -->需要搭配$scope.$apply()使用
        // clearInterval()在上面的方法不适用。
    }])
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
        <input type="button" value="{{ name }}" ng-disabled="isDisabled">
    </div>
</body>
</html>
```

ng-readonly：控制（控件）能否使用（主要用于输入框），让我们可以动态的修改控件状态。

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',["$scope","$interval",function($scope,$interval){
        var Num = 5;
        $scope.name=Num+'秒';
        $scope.isDisabled=true;

        var timer=$interval(function(){
            Num--;
            $scope.name=Num+'秒';

            if(Num == 0){
                $interval.cancel(timer);
                $scope.name='按钮可以使用了';
                $scope.isDisabled=false;
              }
        },1000);

        // setInterval()  -->需要搭配$scope.$apply()使用
        // clearInterval()在上面的方法不适用。
    }])
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
```

```
        <input type="button" value="{{ name }}" ng-disabled="isDisabled"><br>
        <input type="text" value="{{ name }}" ng-disabled="isDisabled"><br>
        <!-- disabled和readonly有一些样式和特性上的差异 -->
        <input type="text" value="{{ name }}" ng-readonly="isDisabled">
    </div>
</body>
</html>
```

ng-checked：控制（控件）的选中状态。

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',["$scope","$interval",function($scope,$interval){
        var Num = 5;
        $scope.name=Num+'秒';
        $scope.isDisabled=true;

        var timer=$interval(function(){
            Num--;
            $scope.name=Num+'秒';

            if(Num == 0){
                $interval.cancel(timer);
                $scope.name='按钮可以使用了';
                $scope.isDisabled=false;
              }
        },1000);

        // setInterval()  -->需要搭配$scope.$apply()使用
        // clearInterval()在上面的方法不适用。
    }])
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
        <input type="button" value="{{ name }}" ng-disabled="isDisabled"><br>
        <input type="text" value="{{ name }}" ng-disabled="isDisabled">
        <input type="checkbox" ng-checked="isDisabled">
        <!-- 当isDisabled为true的时候，复选框是选中状态。 -->
        <!-- 当isDisabled为false的时候，复选框是未选中状态。 -->
    </div>
</body>
</html>
```

ng-value：增加用户体验。

如果我么用的是value=“{{name}}”的话，当我们网络不好的时候会在页面中显示出{{name}}。在解析中。。。

当我们用的是ng-value=“name”的话。页面中就不会显示出表达式的样式。

ng-bind：相当于{{ name }}

案例

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

```
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',["$scope",function($scope){
        $scope.name='hi';

     // alert(1);
    }])
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
        <!-- <p>{{ name }}</p> -->
        <!-- 可以增加用户体验，在网速延迟时候不会显示表达式 -->
        <p ng-bind="name"></p>


        <!-- <p>{{ name }}，{{ name }}</p>
        ng-bind方法不支持多个表达式的解析
        <p ng-bind="name name">错误的</p>  -->
    </div>
</body>
</html>
```

ng-bind-template：是支持表达式的，所以可以支持多个表达式

```
<p ng-bind-template="{{name}}，{{name}}"></p>
        <!-- ng-bind-template是支持表达式的，所以可以支持多个表达式 -->
```

ng-cloak：控制css让当前标签在加载中时候是隐藏的。加载完成后显示。

```
<div ng-cloak>{{text}}</div>
```

ng-bind-html：可以识别和显示标签

```
 $scope.name='<h1>你好</h1>';

<p ng-bind-html="name"></p>
```

<!-- 可以识别和显示标签，但使用率很低。所以需要单独添加模块angular-sanitize.min.js -->

ng-non-bindable：让我的表达式不被解析

```
<div ng-non-bindable>{{ name }}</div>
```

ng-class

```
<p ng-class="{red:true,blue:true}">{{ name }}</p>
        <!-- 用class改变样式 -->
```

ng-style

```
<p ng-style="{background:'blue',color:'red'}">{{ name }}</p>
        <!-- 行内的样式 -->
```

案例

```
<!DOCTYPE html>
```

```
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <style>
        .red{
            background: red;
        }
        .blue{
            color: blue;
        }
    </style>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',["$scope",function($scope){
        $scope.name='hi';
        $scope.class1="{red:true,blue:true}"
        $scope.style="{background:'blue',color:'red'}";
    }])
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
        <!-- <p ng-class="{red:true,blue:true}">{{ name }}</p> -->
        <!-- 用class改变样式 -->
        <p ng-class="{{class1}}">{{ name }}</p>
        <!-- ng-class的好处就是把相应的样式放入数据，可进行操作。 -->

        <!-- <p ng-style="{background:'blue',color:'red'}">{{ name }}</p> -->
        <!-- 行内的样式 -->

        <!-- <p ng-style="{{ style }}">{{ name }}</p> -->
        <!-- ng-style的好处就是把相应的样式放入数据，可进行操作。 -->
    </div>
</body>
</html>
```

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <style>
        .red{
            background: red;
        }
        .blue{
            color: blue;
        }
    </style>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',["$scope",function($scope){
        $scope.name='hi';

        $scope.style="{background:'blue',color:'red'}";
    }])
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
```

```
        <!-- <p ng-class="{red:true,blue:true}">{{ name }}</p> -->
        <!-- 用class改变样式 -->
        <!-- <p ng-style="{background:'blue',color:'red'}">{{ name }}</p> -->
        <!-- 行内的样式 -->

        <p ng-style="{{ style }}">{{ name }}</p>
        <!-- ng-style的好处就是把相应的样式放入数据，可进行操作。 -->
    </div>
</body>
</html>
```

ng-href

```
$scope.url='http://www.baidu.com';

<a ng-href="{{ url }}">百度</a>
```

ng-src

```
$scope.url1='1.jpg';

<img src="{{ url1 }}" alt="">
```

ng-attr-添加属性

```
<a ng-href="{{ url }}" ng-attr-title="{{name}}">百度</a>
```

ng-show：显示

ng-hide：隐藏

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="js/angular.min.js"></script>
    <script>
    var m1 = angular.module('myApp',[]);
    m1.controller('Aaa',["$scope",function($scope){
        $scope.name='请阅读相关数据！';
        $scope.show = true;
    }])
    </script>
</head>
<body>
    <div ng-controller='Aaa'>
        <input type="checkbox" ng-model="show">
        <!-- <div ng-show="show">{{ name }}</div> -->
        <!-- 通过改变标签的display，控制样式的显示隐藏true是显示、false是隐藏 -->
        <div ng-hide="show">{{ name }}</div>
        <!-- 和ng-show相反的操作 -->

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

ng-if

```
<div ng-if="show">{{ name }}</div>
        <!-- 通过添加和删除元素实现操作，并不是改变样式。 -->
```

ng-swtich

–on

–default

–when

```
<div ng-switch on="show">
        <span ng-switch-default>默认显示的效果</span>
        <span ng-switch-when="false">切换后显示的效果</span>
 </div>
```

ng-open：控制部分标签的显示、隐藏

案例

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <script src="js/angular.min.js"></script>
   <script>
   var m1 = angular.module('myApp',[]);
   m1.controller('Aaa',["$scope",function($scope){
      $scope.name='请阅读相关数据！';
      $scope.show = true;
   }])
   </script>
</head>
<body>
   <div ng-controller='Aaa'>
      <input type="checkbox" ng-model="show">

      <div ng-switch on="show">
         <span ng-switch-default>默认显示的效果</span>
         <span ng-switch-when="false">切换后显示的效果</span>
         <details ng-open="show">
            <summary>Copyright 2011.</summary>
            <p>All pages and graphics on this web site are the property of W3School.</p>
         </details>
      </div>
   </div>
</body>
</html>
```

需要搭配H5的标签<details>下面是w3c中的样式说明

```
<details ng-open="show">
      <summary>Copyright 2011.</summary>
      <p>All pages and graphics on this web site are the property of W3School.</p>
 </details>
```