* HTML파일에 AngularJS 적용
  + angular.js파일을 불러오는 script element를 추가함으로써 적용
  + AngularJS는 하나 이상의 module로 구성
    - angular.module method를 호출하여 module 생성
      * 생성할 module name, 필요한 추가 module array를 parameter로 호출
      * var totoApp = **angular.module(“todoApp”, [])**
        + application module name에 App을 첨부하는 명명 관례를 사용해 todoApp이라는 module을 생성
        + 빈 배열을 두번째 parameter로 하여 추가 모듈이 필요 없음을 AngularJS에게 알린다(dependency parameter를 빼면 에러 발생)
  + ng-app attribute를 사용하여 AngularJS가 모듈을 어떻게 적용해야하는지를 알려준다
    - ng-app attribute는 AngularJS가 컴파일하고 처리해야할 모듈이 있음을 알려주는 역할을 한다.
      * 자바스크립트 프레임워크로 AngularJS만 사용하는 경우 관례상 ng-app attribute를 html element에 적용한다
      * AngularJS를 Jquery 같은 다른 기술과 함께 사용하는 경우에는 ng-app attribute를 문서 내 element에 적용함으로써 AngularJS 앱이 차지하는 범위를 제한할 수 있다
    - <html **ng-app=”todoApp”**></html>
  + AngularJS는 새 엘리먼트, 어트리뷰트, 클래스 및 특수 주석을 추가함으로써 HTML을 확장한다.
    - AngularJS 라이브러리는 동적으로 문서 내 HTML을 파싱해 추가 요소의 위치를 파악하고 처리하여 애플리케이션 생성
    - AngularJS 라이브러리는 **브라우저가 컨텐츠를 로드할 때**, HTML 엘리먼트를 해석하고, 표준 DOM API 및 자바스크립트 기능을 사용해 엘리먼트를 추가 또는 제거하고, 이벤트 핸들러 등을 설정
    - AngularJS 개발 과정에서는 명시적인 컴파일 단계가 없고 HTML과 자바스크립트 파일을 수정한 후 브라우저로 로드하는 과정만 있다.
* 데이터 모델 생성
  + AngularJS는 MVC 패턴을 지원한다.
    - 데이터를 데이터 표현 로직과 분리
  + Model
    - application의 데이터
    - 데이터의 저장 및 조회와 관련한 로직 포함
  + Controller
    - 데이터를 처리하는 로직
    - 모델과 뷰 사이에 위치하며 둘을 서로 연결
  + View
    - 데이터를 보여주는 로직
    - 데이터를 formatting하고 사용자에게 보여주는 로직
* 컨트롤러 생성
  + View를 지원하는 데 필요한 비즈니스 로직을 정의
  + 데이터를 처리하는 로직
  + 모델과 뷰 사이에 위치하며 둘을 서로 연결
  + Controller는 사용자 상호작용에 반응하며, 모델 내 데이터를 업데이트하고, 뷰에서 요청한 데이터를 뷰에게 제공
  + angular.module을 호출해 반환되는 Module객체에서 controller method를 호출해 생성
    - 관례상 컨트롤러의 이름은 <이름>Ctrl로 지정(<이름>은 애플리케이션에서 컨트롤러가 하는 역할을 잘 나타내는 이름으로 한다)
    - var todoApp = angular.module(“todoApp”, []);
    - **todoApp.controller(“ToDoCtrl”, function($scope){ $scope.todo = model; });**
    - <body **ng-controller=”ToDoCtrl”**></body>
  + **$scope**
    - View가 항상 전체 모델에 접근할 수 있는 것은 적절하지 않으므로 컨트롤러를 사용하 뷰가 접근할 수 있는 데이터영역을 선택한다
    - **'$'** 로 시작하는 변수명은 AngularJS에서 제공하는 내장 기능을 나타낸다. 주로 내장 **서비스**를 나타냄
    - 서비스는 자체 컴포넌트로 여러 컨트롤러에게 기능을 제공하는 일을 한다
    - $scope은 특수 변수로서, 뷰에게 데이터 및 기능을 노출하는 데 사용한다.
    - $scope.todo = model; → $scope 서비스 객체에 todo라는 속성을 정의하고 model을 대입
  + **ng-controller** attribute 사용하여 컨트롤러가 책임질 HTML 문서 영역 지정
* 뷰 생성
  + 뷰는 컨트롤러에서 제공하는 데이터를 브라우저에서 보여줄 컨텐츠를 생성하게 하는 annotation HTML element와 결합함으로써 생성
  + **Data binding** annotation을 사용
    - **{{~}}**를 사용해 data binding 표현식을 지정
    - 표현식 안의 내용은 자바스크립트에 의해 해석
    - 컨트롤러의 $scope에 지정된 데이터 및 함수로 제한된다
    - AngularJS는 문서 내 HTML을 컴파일하고 ng-controller attribute를 찾은 후, ng-controller로 연결된 함수를 호출해 뷰를 생성하는데 사용할 $scope를 설정한다. 각 데이터 바인딩 표현식을 만날 때마다 AngularJS는 지정된 값을 $scope 객체에서 검사한 후 해당 값을 HTML 문서에 삽입한다.
    - 표현식은 데이터 값을 포여주기 적절하게끔 간단히 수정하는데만 사용해야한다.
    - Data binding은 복잡한 로직을 수행하거나 모델을 조작하는데 사용해서는 안된다.
  + **Directive** 사용
    - 표현식은 Directive와 함께 사용할 수 있다.
    - Directive는 AngularJS에게 컨텐츠를 어떻게 처리해야 할지 알려준다
    - ex :<tr **ng-repeat=”item in todo.items”**><td>{{item.action}}</td></tr> → tr element 및 포함된 element들을 todo.items배열 내 객체별로 생성하고, 배열 내 각 객체를 item이라는 변수에 대입
    - Directive 종류
      * **ng-repeat=”<이름> in <컬렉션>”** → 컬렉션을 차례로 순회하면서 element를 <이름>에 대입한다.
      * **ng-hide=”표현식”** → attribute로 지정된 표현식이 true일 경우, 이 디렉티브가 적용된 엘리먼트(및 컨텐츠 엘리먼트)를 숨겨준다.
      * **ng-class** → 동적으로 클래스명을 바꿀 수 있다.(동적으로 CSS class들을 html에 적용 시킬 수 있다.)
      * **ng-model** → Controller scope 내에서 동적으로 속성을 생성해주고, 사용자 입력을 처리하는 데 사용할 수 있는 동적인 모델 속성을 생성해준다.
      * **ng-click=”표현식”** → click event가 일어날 때 표현식을 실행하는 핸들러를 설정
  + **양방향 바인딩**
    - 단방향 바인딩 : 모델에서 값을 가져와 템플릿 내 엘리먼트를 채우는데 사용
    - 양방향 바인딩
      * 모델은 엘리먼트를 생성하는 데 사용될 뿐 아니라, 엘리먼트 내 변화가 생기면 모델 또한 같이 바뀌게 된다.
      * 양방향 바인딩은 사용자 입력을 받는 엘리먼트에 적용할 수 있다.
      * **ng-model** attribute
        + AngularJS에서 입력 엘리먼트와 데이터 객체의 done 속성 사이에 양방향 바인딩을 적용하게끔 한다.
        + <input type=”checkbox” **ng-model=”item.done”**/>
* 컨트롤러 동작 구현 및 활용
  + 컨트롤러는 scope의 동작을 정의한다.
    - 동작
      * application의 비즈니스 로직을 구현하기 위해 모델 내 데이터에 수행하는 기능
      * 컨트롤러에서 정의한 동작은 데이터를 사용자에게 보여줄 수 있게 뷰를 지원하며, 사용자 상호작용을 기반으로 모델을 업데이트한다.
    - 동작은 컨트롤러 함수로 전달된 $scope 객체에 함수를 추가함으로써 정의
    - $scope 객체에 함수를 첨부하는 데 사용한 속성명은 동작명으로 사용되며, 뷰를 형성하는 HTML 엘리먼트에 컨트롤러를 적용하는 ng-comtroller attribute scope 내에서 호출할 수 있다.('()'를 사용해 호출)
    - 동작으로는 객체를 인자로 넘겨줄 수도 있다.
* 사용자 상호작용 반응
  + 실제 프로젝트에서는 뷰를 위해 데이터를 가져오고 준비하는 동작과 사용자 상호작용에 반응하고 모델을 업데이트하는 동작이 여러개로 분리되어 있는 게 보통이다.
  + AngularJS가 HTML을 컴파일하다 directive를 만나면, AngularJS는 이벤트 핸들러 코드가 엘리먼트와 분리되게끔 핸들러를 설정해준다.
    - <input type=”button” **ng-click=”addNewItem(actoinText)”** value=”add”/>
    - AngularJS가 이벤트 핸들러 코드를 엘리먼트와 분리되게 해준다
* 모델 데이터의 필터 및 정렬
  + 동작을 구현하지 않고도 뷰에서 보여줄 모델의 데이터를 효과적으로 준비할 수 있다.
  + 동작을 사용하는 것보다 좀 더 범용적인 용도로 사용하고, 애플리케이션 전역에서 재사용하는 특징이 있다.
  + 필터링은 데이터 모델의 어느 부분에나 적용할 수 있다.
  + Filter 종류
    - **filter**
      * 설정된 기준에 따라 객체를 선택
    - **orderBy**
      * 데이터를 정렬
      * 정렬에 사용하려는 속성을 리터럴값으로 지정
      * 리터럴 값으로 지정하지 않으면 scope 속성을 찾음
        + 기본적으로 AngularJS에서는 모든 것을 $scope에 의해 정의된 속성으로 간주하므로 리터럴 값으로 사용하지 않으면 $scope 속성을 찾는다
        + 상수를 지정해야할 때는 리터럴 값 사용
  + 커스텀 필터
    - AngularJS 모듈 객체에서 정의한 filter() method로 filter factory를 생성할 수 있다.
    - Filter factory → 데이터 객체 셋을 필터링하는 데 사용할 함수를 반환
      * angular-module.**filter(“filter-name”, function(){return function(items){}})**
      * items parameter → AngularJS에서 제공하며, 필터링할 객체 셋
      * items parameter외에 parameter 사용가능
        + filter(“filter-name”, function(){return function(items, showComplete){}})
    - 사용
      * **custom-filter:전달인자**
* Ajax를 통한 데이터 수신
  + Ajax 요청을 통해 데이터를 받을 수 있다.
  + Angular-module.**run(function($http){$http.get(“url”).success(function(data){})})**
    - AngularJS 모듈 객체에 정의된 run method에 호출 코드 추가
    - run method는 AngularJS가 **초기 설정을 마치는 시점에 한 번만 실행할 함수**를 parameter로 받는다
    - $http 서비스 객체를 parameter로 지정하여 AngularJS에게 $http 서비스를 사용할 것을 알린다.(dependency injection)
    - $http 서비스 객체
      * 저수준 Ajax 요청에 대한 접근 기능 제공
      * RESTful 웹 서비스에서 사용하는 $resource와 비교할 때 실제로 그렇게 저수준은 아님
    - get() method의 결과는 **Promise 객체**로서, 이 객체는 미래에 완료될 일을 나타내는 데 사용된다.
    - 서버에서 조회한 JSON 데이터는 파싱을 거쳐 자바스크립트 객체가 되고, success 함수의 data parameter로 넘어온다
  + **Dependency Injection**
    - parameter를 사용해 필요한 기능을 AngularJS에게 알려주는 방식