架构：

架构图

8组成：

Module 组成：5部分。

根模块

Module 组成：5部分。

显示数据

3方式

插值:{{var}}

NgFor: <li \*ngFor = “let todo of todoes”> {{todo.desc}} </li>

用构造函数的参数直接定义属性。

ngif: <p \*ngIf = “heros.length > 3” > There are many heros!</p>

用户输入

绑定到用户输入事件： <button (click) = “onClickMe()”> Click me </button>

通过$event对象取得用户输入：

Html:

<input (keyup) = “onKey($event)”>

<p> {{value}} </p>

Component:

onKey(event: keyboardEvent){

this.values + = event.target.value+’|’;}

从模板引用变量中获得用户输入：

Html：

<input #box (keyup) = “onKey(box.value)”>

按键事件过滤：

Html：

<input #box (keyup.enter) = “onKey(box.value)”

(blur) = “onKey(box.value)”>

表单

[(ngModel)]：

Component.ts:

import {FormsModule} from ‘@angular/forms’;

Html:

<form #heroForm = “ngForm”>

name属性的用途是有效性验证和对表单元素变更进行追踪。在表单中使用[(ngModel)]时，必须要定义name attribute。

通过ngModel跟踪修改状态与有效性验证： ng-untouched/ng-touched; ng-dirty/ng-pristine;ng-valid/ng-invalid

<button type=”button” class=”btn btn-default” (click)=”newHero(); heroForm.reset()”>

使用ngSubmit 提交表单。<form (ngSubmit) = “onSubmit()” #heroForm=’ngForm’>

依赖注入

Angular的依赖注入

配置注入器：在angular的启动过程中就会自动创建一个应用级注入器。要么在NgModule中注册提供商，要么在应用组件中提供。

在NgModule中注册提供商

Providers:[

userService,

{provide: APP\_CONFIG, useValue:HERO\_DI\_CONFIG}

]

创建被注册日志服务

注入器提供商们

provider类和一个提供商的字面量 [{provide: Logger, useClass: Logger}]第一个属性是token,是key，用于定位以来至和注册提供商。第二个属性是一个提供商定义对象，知道如何创建依赖值。

备选的类provider,

带依赖的类provider

别名类provider: [NewLogger, {provide: Logger, useExisting:NewLogger}]

值提供商：[{provide: Logger, useValue:silentLogger}]

工厂提供商

依赖注入令牌

可选依赖

直接使用注入器

模板语法

模板中的HTML： <script>会被忽略

插值表达式

模板表达式。 有一些JS的合法表达式会被禁止（=，+=，-=，new，++，--）不支持位运算|和&。模板表达式中的|和？是模板表达式定义的。

表达式上下文： 模板输入变量（let hero）和模板引用变量(#heroInput如果要引用的变量名存在于一个以上的命名空间，那么模板变量时最优先的。例如组件和模板中同时存在同名变量，优先使用模板变量。模板表达式不能引用全局命名空间中的任何东西。只能引用表达式上下文的成员。

表达式指南：

没有可见的副作用

执行迅速

非常简单

幂等性：连续调用两次应该返回同一个值

模板语句：支持=和表达式链；但是new，++，--，+=，-=，|，&和模板表达式运算符

语句上下文

没有可见的副作用

绑定

属性绑定(property binding) <img [src] = “heroImageUrl”>

单向输入

绑定目标