# gulp

第1天课堂笔记

复习

第一天

Ng-app定义app指令

值表示app的名称

Ng-init 初始化数据

定义变量不用var关键字，多个关键之间用;隔开

绑定

插值 {{key}}

绑定 ng-bind

获取应用程序

Angular.module

第一个参数表示app的名称

第二个参数表示依赖集合（使用的插件要在依赖集合中引入）

控制器 controller

第一个参数表示控制器的名称（要通过ng-controller指令在dom定义的）

第二个参数表示控制器工厂方法

默认没有参数，使用什么参数就要注入什么参数

作用域空对象

参数注入

参数的名称不要写错

Ng-click

与vue比较

相同点

都可以使用$event参数，使用事件对象

不同点

定义事件

Angular ng-

Vue v-on:

回调函数参数集合

Angular必须加上

Vue 可以不用加

回调函数定义

Angular 作用域上

Vue 定义在methods上

回调函数的作用域

Angular $scope

vue vue实例化对象

回调函数参数

Angular 与传递一致

Vue 但是如果没有参数集合，默认有一个参数就是事件对象

修改数据

Angular

This

$scope

Vue

This

Ng-show 是否显示一个元素

True显示

False隐藏

Ng-hide 是否隐藏一个元素

True隐藏

False显示

这两个指令都是通过添加ng-hide类来实现显隐的

Vue是通过style=”display: \*\*” 来显隐的

插值表达式，

在angular中我们可以使用插值表达

插值过滤器

字符串相关

uppercase 大写

lowercase 小写

limitTo 截取

第一个参数表示截取场地

第二个参数表示截取位置

也可以适用于数组

Currency 过滤器

第一个参数前面符号

第二个参数表示小数位保留的位数

Json 过滤器

转化成json字符串

Number 过滤器

转化成科学计数法

参数表示保留小数位数

Date过滤器

参数格式化的模板

y 年

M 月

d 天

h 12进制小时

H 24进制小时

m 分

s 表示

sss 毫秒

E 星期

orderBy过滤器

排序字段

数字比较大小

字符串字母编码排序

第二个参数表示是否倒序

Filter 过滤器

第一个过滤字段

字符串过滤器字段名称

变量

方法 不要执行，返回值是判断的条件

方法如果执行了相当于使用他的返回值

自定义过滤器

Filter方法

第一个参数表示过滤器名称

第二个参数过滤工厂函数

没有参数

作用域空对象

只能执行一次

返回值是真正的过滤函数

第一个参数表示处理的数据

从第二个参数开始表示传递的参数

作用域是全局作用域

返回值表示输出的数据

在插值中，我们可以使用多个过滤器，通过|分隔，

前一个过滤器的输出作为后一个过滤器的输入

第二天

表单验证

四个属性

$dirty 是否修改过

$pristine 是否没有输入过

$valid 是否合法

$invalid 是否不合法

表单元素添加name属性，并且会在作用域上创建一个相应的变量

验证指令

必填项

Required ng-required

相同点

1 都是在没有输入就开始验证

2 都是做必填项验证

不同点

Required是h5的属性，ng-required是angular指令

Require不需要赋值，ng-required必须要赋值

长度

Minlength maxlength ng-minlength ng-maxlength

相同点

都是在输入后开始验证

都是做长度验证

不同点

Minlength，maxlength是h5 的属性，ng-minlength，ng-maxlength都是angular定义的指令

正则验证

Pattern ng-pattern

相同点

都是在输入时候验证

都是做正则验证

不同点

Pattern是h5属性，ng-pattern是angular指令

Pattern的值是正则的内容，ng-pattern是完整的正则表达式

Ng-disabled是否禁用

True禁用

False可以使用

Ng-readyonly 是否是只读的

True 只读

False 可以写入

Ng-change 输入内容改变时候触发

无法访问事件对象

必须依赖ng-model

Ng-submit 提交事件

该事件是绑定在form表单上

触发该事件，是form表单元素内部具有submit类型子表单元素

Run方法

在这个方法中可以使用根作用域$rootScope

在应用程序运行时候，会调用这个方法

作用域

基于原型式继承，就近原则

使用的数据是原型链上离他最近的数据

一旦父作用域数据改变，会影响子作用域

$parent 父作用域

$$prevsibling 前一个作用域

$$nextsiblig 后一个作用域

$element

获取控制器容器元素，返回是一个jqlite对象

Ng-href动态改变href属性

Ng-src动态改变src属性

属性值就是普通字符串，因此要使用插值符号使用变量

Ng-class 绑定类

传递一个字符串

传递一个变量

传递一个对象

属性名称表示类的名称

属性值是布尔值

传递数组

数组每一个成员表示一组类

这四组值都可以包含空格，表示多个类，但是，在数组和对象中我们避免这样使用，避免让类之间产生耦合

Ng-style 绑定样式

传递一个对象

传递一个变量（在作用域中定义的对象）

对象属性名称表示css样式名称

对象属性值表示样式的值

Ng-if 判断指令

True创建

False删除

Ng-switch 多分支判断指令

On 判断的条件变量

Ng-switch-when 判断的某一个条件

值是一个字符串

Ng-switch-default 没有值，表示默认情况下

Ng-repeat 循环模板指令

语法 ng-repeat=“item in data”

Item临时循环变量

Data作用域中的数据

创建了6个临时变量

$index 索引值

$first 是否是第一次

$last 是否是最后一次

$middle 是否是中间的

$even 是否是偶数次

$odd 是否是奇数次

Ng-include 异步加载模板

必须搭建服务器

值是js作用域

第三天

自定义指令

第一步在html中引入指令 以-分隔单词

第二步通过directive定义自定义指令

第一个参数表示名称

驼峰式命名

第二个参数工厂方法

作用域全局作用域

执行一次

没有参数

返回值是一个对象

定义指令类型

Restrict 值有四个 ECMA

E element 表示元素类型

C class 表示类类型

M comment注释类型

A attribute 属性类型

Template/templateUrl定义模板

Replace 元素是否被替换

True替换

False不替换（默认值）

Controller定义控制

作用域scope

容器元素内部的模板

False（默认）

父子作用域是同一个

True

父子作用域隔离的

{}

父子作用域隔离

模板中使用的数据是父作用域中的

容器元素外部模板

False

同上

True

同上

{}

同true

当scope是{}时候，为了在子作用域中使用父作用域中的数据，我们可以使用作用域修饰符

@修饰符

第一步在容器元素属性中通过插值符号引入父作用域中数据

第二步在scope中通过@修饰符定义属性数据

特点

单向，只能从父作用域向子作用域传递数据

覆盖关系 父作用域覆盖子作用域

=修饰符

第一步绑定数据不要使用插值符号

第二步在scope中使用=修饰符

特点

双向的

覆盖关系 子作用域覆盖父作用域

Link 编译方法

第一个表示作用域

第二个参数表示元素

Jqlit对象

第三个参数表示属性对象

$attr表示对象属性与元素属性名称 的映射

以data开头的属性会忽略data-

第四个参数表示控制器服务

作用域全局作用域

Compile 表示编译的全过程

第一个参数表示元素

第二个表示属性对象

作用域是指令描述对象

返回值就是link方法

作用域是在这个方法执行完毕之后，link方法执行之前创建的

$watch 监听属性对应数据的变化

第一个参数表示数据名称

第二个参数表示回调方法

第一个表示当前

第二个表示上一个值

第三个作用域

作用域属性对象

Transclude 在模板中引入元素未知内容

第一步在模板中通过ng-transclude指令定义模板容器

第二步设置transclude属性值是true

Require 属性值表示依赖的服务，此时会将该服务添加给link方法的第四个参数

属性值如果是字符串表示一个服务

如果是数组表示多个服务

$timeout 服务表示对setTimeout封装

$interval 表示对setInterval封装

作用域丢失四种解决方法

第一个通过服务

第二个是$digest

第三个是$apply

第四个$apply回调函数

第四天

$http服务

参数是一个对象

Url请求地址

Method请求方法

Data请求数据

Params请求参数

返回值是promise对象

Success方法

回调函数有四个参数

返回数据

状态

函数

请求配置对象

方法返回值是一个原promise对象，因此可以链式调用多个success方法

Then参数是一个对象，对象中data属性表示返回的数据

返回值是一个新的promis对象因此再次调用就没有参数了

Get 是get请求简写方式

地址

配置信息

Post 是post请求简写方式

地址

Data数据

配置信息

自定义服务

面向过程 factory

通过return暴漏接口

不能用this，会被return覆盖

面向对象 service

通过this暴漏接口

不能用return，会覆盖this

在定义服务中我们可以参数注入其他服务，但是不允许双向引用

路由

指令路由

引入插件

定义容器元素 ng-view

引入模块 ngRoute

在配置中引入服务 $routeProvider

Config 是在所有方法执行执行，最先执行的一个方法

$routeProvider 有个when方法

表示地址

路由对象

Template/templateUrl定义模板

Controller定义控制器

值是字符串，一定要在后文中定义这个控制器

值是函数

动态参数 /:key 此时key会添加到路由参数对象中，

我们在控制中注入$routeParams服务使用路由动态参数

Otherwise定义重定向路由

redirectTo属性定义默认路由

路由事件

$routeChangeSuccess

路由事件对象

当前路由对象

前一个路由对象

$location服务

包含路由信息，每一个路由信息都对应一个方法用来获取或者设置路由

例如$$path 对应path方法

如果没有参数表示获取

如果有参数表示设置

Ui路由

引入插件

定义路由容器 ui-view

引入依赖模块

Ui.router

注入$stateProvider服务

State

第一个参数表示状态名称

第二个参数表示路由对象

Url定义地址

Tempalte/templateUrl定义模板

Controller定义控制器

$stateParams使用路由参数

固定路由参数 /key

动态参数 /:key

Query参数 ?key&key2

动态参数类型 /{key:type}

Query参数类型 ?{key1:type}&{key2:type}&key3

Query参数是可有可无的

多个留有容器

通过在页面中ui-view赋值

在路由对象中

通过views定义多个页面

每一个值都是一个对象

可以定义模板，控制器等

子路由

第一步在父路由中定义ui-view指令

第二步在子路由中使用父路由命名空间

第一个参数状态名称中使用

子路由完整路径：父路由地址+子路由地址

事件

$stateChangeSuccess

路由事件对象

当前路由路由对象

当前路由参数对象

上一个路由对象

上一个路由参数对象

gulp -v

npm install -g gulp

# Gulp

Gulp是工程化开发的一个工具

比如我们在公司开发项目，项目中文件有很多，对于文件依赖关系，打包，压缩，代码检测等等，如果仅仅通过我们人手动的处理是有风险的，但是如果用机器去处理这些事情就能屏蔽这些风险，工程化就是用机器去批量的处理文件，例如对js，css，html压缩，对他们的打包，对es6，typescript，less，sass，style语言的编译

Gulp是基于流出来的

例如UNIX中的概念，上一次操作的输出作为下一次操作的输入，比如node中的流

例如我们压缩一个js

第一步，通过gulp获取这个文件，并转化成流（文件名，格式，编码，内容）

第二步，对流的内容进行压缩

第三步，将压缩的内容重新写回一个新的文件

## 获取

|  |
| --- |
| 1. npm install -g gulp 2. // mac linux要加上管理员命令 3. sudo install -g gulp |

查看是否安装成功，我们可以通过gulp -v查看版本号的形式来检测

Gulp建议我们使用本地的包，不要全局环境中包

## 几个网站

中文网

<http://www.gulpjs.com.cn/>

官网

<http://www.gulpjs.com>

插件库

<http://gulpjs.com/plugins/>

Npm包库

https://www.npmjs.com/

Gulp项目开发的第一步要当前目录下建立一个gulpfile.js文件，这个就是gulp的入口文件，当我们执行gulp指令的时候，会搜索这个文件是否存在，如果存在会执行该文件里面的任务，

## api

Gulp简单，只有5个api（task，src， pipe， dest， watch） （run）

Gulp基于任务开发模式，在一个项目，做一件事情就是在完成一项任务，我们可以通过task定义任务

## Task

Task(name, fn)

Name 表示任务名称

Fn 该任务执行函数

参数是一个函数

作用域是gulp对象

因此可以通过this调用其他gulp属性方法

但是gulp建议我们函数式编程，因此不要使用this调用法，而是用gulp调用方式

Task(name, [])

Name表示任务名称

[] 表示依赖的其他 任务

在gulp开发中，跟开发nodejs是一模一样的，

因此使用gulp就要引入这个模块

Require(‘gulp’)

## run

Run 启动任务的

参数表示任务名称

在新版本中不建议我们用run

而是用default任务来启动我们的任务

|  |
| --- |
| 1. gulp.task('show', function () { 2. // console.log(arguments) 3. // console.log(this) 4. // 任务就是输出爱创课堂 5. console.log('爱创课堂123') 6. }) 7. // 可以通过run方法执行这个任务 8. gulp.run('show') 9. gulp.task('default', ['show']) |

读写操作

## src

读文件操作

参数表示文件的路径

作用是将文件读取并转化成一个流

## Pipe

管道方法

参数是处理事情

一定要在src获取的文件后面执行

## Dest

写入文件

参数表示写入的地址

作用就是将流转化成一个文件，结束操作

|  |
| --- |
| 1. gulp.task('dealFile', function () { 2. gulp 3. // 第一步获取文件 4. .src('js/demo.js') 5. // 在管道中处理文件 6. .pipe(gulp.dest('demo')) 7. }) |

## watch

文件监听，监听文件变化并做处理

例如我们在开发中会不停的修改文件，那么当修改完成，我们为了发布，需要再执行一次gulp指令。保存是很频繁的，因此每一次保存执行一次gulp很繁琐，因此gulp提供了一个功能叫watch方法，用来监听文件变化并做相应处理

改变值得就是保存，删除，创建等

Watch(path, fn)

Path监听的文件

Fn表示文件改变时执行的方法

参数是一个事件对象

Type表示更改的方式，changed，added，deleted

Path表示文件绝对路径

Watch(path, [])

Path监听的文件

[] 文件改变时，执行的任务

|  |
| --- |
| 1. gulp.task('watchJs', function () { 2. // 第一步获取文件 3. gulp 4. .src('js/app.js') 5. // 将文件发布 6. .pipe(gulp.dest('abc')) 7. }) |

## Globs语法

解决问题：一条地址只能匹配一个文件的问题的

有点类似我们学过的正则，不过globs语法要比正则简单的多，因为他是专门为匹配路径设计的规则

规则

\* 用来匹配 任意字符串（不包括目录分隔/）

Js/\*.js

匹配 js/a.js js/abc.js

不匹配 js/a/b.js js/a.css

\*\* 用来匹配任意字符串（路径及其子路径），

如果首位有其他字母，他只能用来匹配路径

Demo/\*\*/\*.js => demo/a/b/c/de/a.js

如果后面什么都没有，可以匹配到文件

Demo/\*\* => demo/ab/c.js

? 匹配零或一个字符

Ab?c.js 匹配 Abc.js Ac.js 不匹配adc.js

[] 匹配字符集

a[bcd]ef.js 匹配 abef.js acef.js 不匹配 abcdef.js axef.js

[^] 不匹配的字符集

a[^bcd]ef.js 匹配 axef.js 不匹配 abef.js

可以应在上诉5个api中使用path参数的部分

|  |
| --- |
| 1. gulp.task('moveJs', function () { 2. // 获取js 3. gulp 4. .src('js/\*\*/\*.js') 5. // .src('js/\*\*') 6. // 发布 7. .pipe(gulp.dest('abc')) 8. }) |

# 插件

Gulp工具包只实现对文件的获取，发布以及监听，没有对于文件如何处理做说明，所以打包，压缩，重命名等操作没有实现，这些功能被其他人通过插件的方式实现了

Gulp优势就是轻量级，容易学习，  
 把大量业务放在插件中去实现，因此市面上有好多的插件，

官方没有定义指定插件，因此我们在实现某一个需求时候，我们要自己去插件库中寻找，如果找不到只能去社区，论坛提问

## 重命名插件

Gulp-rename

安装 （sudo） npm install -g gulp-rename

使用插件简单

第一步，引入插件模块

第二步， 在pipe管道中用插件处理文件

接收一个参数表示重命名文件的新的名称

|  |
| --- |
| 1. gulp.task('renameJs', function () { 2. // 获取文件 3. gulp.src('js/app.js') 4. // 先重命名再发布 5. // 第二步 在管道中处理文件 6. // .pipe(rename('起个名字好难')) 7. .pipe(rename('demo.txt')) 8. // 发布 9. .pipe(gulp.dest('demo')) 10. }) |

## 压缩js插件

Gulp-uglify

使用方式跟rename很像

第一步，引入插件模块

第二步，在pipe管道中处理压缩文件

|  |
| --- |
| 1. gulp.task('uglifyJs', function () { 2. // 获取文件 3. gulp.src('js/\*.js') 4. // 在管道中压缩js 5. .pipe(uglify()) 6. // 发布文件 7. .pipe(gulp.dest('dest')) 8. }) |

## 压缩css

Gulp-minify-css

第一步 引入插件

第二步 在管道中压缩css文件

|  |
| --- |
| 1. gulp.task('minifyCss', function () { 2. // 获取文件 3. gulp.src('css/\*.css') 4. // 压缩css 5. .pipe(minifyCss()) 6. // 发布到dest目录 7. .pipe(gulp.dest('dest')) 8. }) |

## 压缩html

Gulp-minify-html

第一步 引入插件

第二步 在管道中压缩html

|  |
| --- |
| 1. gulp.task('minifyHTML', function () { 2. // 获取文件 3. gulp.src('index.html') 4. // 压缩 5. .pipe(minifyHtml()) 6. // 发布 7. .pipe(gulp.dest('dest')) 8. }) |

## Backbone项目实战

将backbone项目内的内容发不到一个上线环境 demo

Data目录是接口文件，我们要发布到demo/data

Img目录是图片文件，我们要发布到demo/img

Static目录里面有css文件，也有js文件，我们要将他们压缩，发布到demo/static

Index.html文件 压缩发布到demo/index.html

App.js 文件 直接发不到demo目录下

在使用globs语法时候，获取文件是，前面固定的目录，在发布时候一定要写全

|  |
| --- |
| 1. gulp 2. .src('12/static/\*\*/\*.js') 3. // 压缩 4. .pipe(uglify()) 5. .pipe(gulp.dest('demo/static')) |

## 文件打包插件

Gulp-concat

使用方式跟rename插件一样

第一步引入插件

第二步打包文件（参数表示打包后的文件名称）

|  |
| --- |
| 1. gulp.task('concat', function () { 2. // 获取文件 3. gulp 4. .src('js/\*.js') 5. // 压缩 6. .pipe(uglify()) 7. // 打包 8. .pipe(concat('ab.js')) 9. // 发布 10. .pipe(gulp.dest('js')) 11. }) |

## Js校验

Gulp-jslint

做js规范校验的

避免我们写的代码产生bug

第一步引入插件

第二步打包文件

|  |
| --- |
| 1. gulp.task('jslint', function () { 2. // 校验js目录下所有js文件 3. gulp.src('js/\*.js') 4. // 校验 5. .pipe(jslint()) 6. .pipe(gulp.dest('dest')) 7. }) |

## 图片压缩插件

Gulp-imagemin

使用方式同上

Png图片压缩收益最大的

|  |
| --- |
| 1. gulp.task('imagemin', function () { 2. // 获取图片，压缩图片 3. gulp.src('img/\*.\*') 4. .pipe(imagemin()) 5. .pipe(gulp.dest('dest')) 6. }) |

# 作业

回去安装fis3

|  |
| --- |
| 1. sudo npm install -g fis3 // mac 2. npm install -g fis3 // win |