**Npm audit fix --force**

Host on Github,

repository name must be [**https://yiditheawe1.github.io/**](https://yiditheawe1.github.io/)

git bash: cd repo directory

**echo "Hello World" > index.html**

**add,commit**

**git push -u origin main**

**等若干时间才会显示**

1.Function Statement VS expression

* 1. Expression=assignment
     1. Request.onreadystatechange= handleResponse;

此处将function handleResponse赋值给onreadystatechange,所以末尾没有括号和参数。带上括号参数则是将function return value赋值

带参数handleResponse(request, r esponseHandler)为执行function

1. Name Space closure
2. IIFE : runs as soon as its defined
3. Arrow function : redefined scoping of ‘this’ keyword(static this)
   1. Normal function defines ‘this’ at where it is called. Called at global, ‘this’ means global
   2. Arrow function defines ‘this’ to be where arrow function is defined静态不变
   3. Arrow function is not constractor:
      1. Let person=(name,age)=>{this.name=name;this.age=age;}
      2. let me= new Person(“a”,30)报错，因为箭头函数不是constructor
   4. 不能使用arguments变量

Let fn=()=>{console.log(arguments)} fn(1,2,3)报错

* 1. 简写：

1. Call back 中文：function a(callback){//body…}先执行body再回到参数执行callback
2. Ajax lecture 57
3. Promise: arrow func，有两个参数(resolve, reject),func定义 resolve reject内容。 Promise带连个后续方法arrow func：.then, .catch,分别用回调函数处理resolve reject || avoid callback hell（创建new promise obj时 resolve、reject已经执行， 后面代码目的是处理promise返回信息）

Text

Description automatically generated

1. Fetch()
2. Node js
3. Containerization
4. Axios一个ajax 库
5. 函数对象方法： call apply ： 通过参数对象修改函数的this
   1. Add.call()==Add.apply():执行add
   2. 调用call和apply可以将参数对象指定为第一个参数：

例1：var obj={}; Add.call(obj)使add的this变成obj，而不再是window

例2：obj={name:”obj”, sayName:function(){alert(this.name)}}; var obj2={name:“obj2”}

obj.sayName.apply(obj2) =>obj2 因为参数对象为obj2

* 1. 区别
     1. Call()可以将实参对象在之后依次传递
        1. 例：fun.apply(obj,2,3):fun需要参数a b。传obj为改this
     2. Apply()需要将实参封装到数组中统一传递
        1. Fun.apply(obj,[2,3])
  2. 函数调用：fun(); this—window obj.fun()- obj

Text, letter

Description automatically generated

1. 构造函数：
   1. 工厂方法创建对象，构造函数都是object，所以生成对象都属object类，导致无法区分不同类型对象

Function createPerson(name, age){

Var obj=new Object();

Obj.name=name; obj.age=age;

Obj.SayName=function(){alert(this.name);}

Return obj;

}

Var jack=createPerson(“jack”,29);

* 1. 构造函数(类) 调用时要加new

流程：  
1.new立刻新建对象2.函数新建对象以this引用3.逐行执行函数代码4.返回新建对象

function Person(name){

this.name=name; //不加this name是window的

} var per=new Person(“jack”)

* 1. 用类代替构造函数

Class Person{

Constructor(name){this.name=name}

} let p=new Person(“bob”)

1. ;大部分情况加不加无所谓。必要在1.小括号(匿名fuc/IIFE)之前2.中括号(数组)前
2. Var vs let vs const:\
   1. Var: global scope / function scope
   2. Let: block scope: {}
   3. Const：aslo block scope, and it’s static, cannot change after declaration
3. 同源策略 前（网页资源）后（目标请求资源）端 相同的协议 域名 端口。京东前后端同源

Ajax默认遵循同源策略

服务器增加就会形成跨域

1. 闭包：当前套内部函数引用嵌套外部函数的变量，则生成闭包

**Angualr 环境搭建运行:**

1.安装node js， （npm）验证node/npm -v

2.安装脚手架 npm install -g @angular/cli 验证ng v

Ng/npm install @angular ng add

ng add @ngrx/effects@latest @ngrx/effects npm uninstall @ngrx/effects

ng add @ngrx/store-devtools

-

3.cd到desired目录创建新的项目,同时安装依赖 ng new my-app11

4.运行项目 ng serve –open

**Angular目录结构**

E2e:端到端（不见）

Node\_modules:package.json 里面的dependencies

Src/app: modules

项目入口 main.ts

核心文根模块件App.module.ts：告诉angular如何组装应用

组件组成：component ts(业务逻辑) html() css()

组件间可以相互挂载， 最终挂载到根组件

创建组件：

用cli，结束项目，cd项目ng g component components/news

App.module.ts会自动引入新创之组件，并配置在装饰器中

引用组件：

在该组件.html中编辑内容, 在根组件html中插入引用标签：<app-news></ app-news >

也可以在子组件html中相互插入引用标签，嵌套引用

绑定数据：在组件ts中(expor class)定义,不用var; 在组件html中绑定:{{title}}

**Angular组件 及里面的模板**

1.组件ts export中定义数据/属性，在构造函数中赋值/变值

2.在组件html中绑定数据{{userInfo.age}}

3.循环数组 用\*ngFor， \*ngIf, \*ngswitch ----标签中[]:绑定属性， ():绑定事件

Ngclass ngStyle:动态改变class 通过flag

4. Pipe:在模板(html{{}})中调用方法(date格式)

**Pipe Usage 学习！！同步异步相关吗？**

5.MVVM: Html值影响ts数据值；同时ts数据值的改变会影响html呈现值

[(ngModel)]=”keywords”

双向数据邦迪必须引入form module

Strong标签

Dom定义

Css margin居中

Form

**JavaScript window object localstorage:**

将数据以key value pairs存储与浏览器domain & protocol

页面刷新时会触发组件中OnInit()生命周期函数！

**Service:组件间所有公用方法的repository,**

**ng g service mydir/new-service**

组件可以调用Service，Service不能调组件。 组件不可互调，但有父子继承，可以相互引用，之后父组件可以调用子组件的方法。Service间可以互调

**Angular获取Dom节点：**

不能在ngOnInit函数中直接捕获节点，如果节点标签带有ng指令。ngOnInit只是组件和指令初始化完成而非加载完成。要等dom加载完成再捕获：

1. jQuery$(function(){})中捕获
2. ngAfterViewInit ():void{}

Native element 指的是浏览器中的dom元素

**ViewChild(获取dom节点、获取组件(引入子组件时 子组件亦是节点)):**

1. html模板中tag起名字#
2. ts中import view child

3.在ts 类中用装饰器获取节点 @ViewChild('mybox') myBox:any

1. ngafterviewInit生命周期函数里获取dom: this.mybox.nativeElement

@Input: 父组件传给子组件，子组件调用父组件

@viewChild父组件获取子组件方法

Graphical user interface, text

Description automatically generated

**父子组件传值：@input(父组件传值(方法/对象/变量)给子组件), viewChild(子组件传值给父组件)**

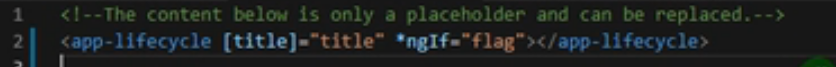
Local storage, service可以用以非父子组件通信

子组件通过@output事件驱动出发父组件的方法

**生命周期函数(钩子方法)：lifecycle-hooks:8个方法：组件创建更新销毁时触发**

1. ngOnChanges() : 父子组件传值中，改变数据时
2. ngOnInit():初始化指令组件，ngOnChanges()完成后使用 只一次。一般用以请求数据
3. ngDoCheck(): 内含自定义操作，例如查看数据是否改变而相应设置action
4. ngAfterContentInit():组件渲染之后
5. NgAfterContentChecked() check中包含自定义行为
6. ngAfterViewInit():视图加载完成 dom操作
7. ngAgterViewChecked()
8. ngOnDestroy()：组件销毁时触发(其中可执行销毁前保存当前数据的actions)

挂载/卸载生命周期函数(\*ngIf, flag为父组件中的bool)



Text

Description automatically generated

RXJS:异步编程

捕获异步方法数据：

方法1： 以指针(不加参数)cb传回调函数，用以承接异步timeout数据

 Graphical user interface, text

Description automatically generated

方法2：Promise

Text

Description automatically generated Text

Description automatically generatedText

Description automatically generated

方法3：RXJS（非常像promise标准，但rxjs是一个库）

1. 引入rxjs Observable对象

Text

Description automatically generated Text

Description automatically generated

Observable~=Promise;

next~=resolve；

Observer~=then中的回调函数，用next接受promise内部发出的数据；

subscirbe~=then链接promise和then的回调

RXJS功能很多，包括中途撤回，发射多值，工具函数等

**Dependency Injection**

Instead of creating, DI makes constructor consumes dependenciesGraphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

**Bundler:**

A screenshot of a computer

Description automatically generated with medium confidence

目的：得到app.js, 使所有的code**be production-ready loadable in the browser**.

**Functional programming:**

**Decomposing** a problem into small and reusable functions that take some inputs and return result

**Pros**: make multiple function calls in parallel and take advantage of multiple CPU cores

JS multi- paradigm, 并非纯fp，所以 有caveats

**Higher order function**: functions that takes a function argument or returns a function

**Function composition & currying a function** :

Text

Description automatically generated Text

Description automatically generated

以下Pipe返回函数并且方法中参数必须都是function而不能是function执行结果(str/num..)。由于warp(“div”)返回一个箭头函数所以ok。

Trim返回箭头函数作为toLowerCase的input，同理给wrap。 “div”作为第二个参数传入。

调用Transform(input)时， input作为第一个参数传入

Text

Description automatically generated

**Update obj w/o mutation(return new obj):**

1.Const person={name:”John”} mutation: person.name=”Joe”

Not mutation:Const person2=Object.assign({},person,{name:”Joe”, age:30})

2.const person2={…person, name:”Joe”} (spread operator 复制，后面更改值。未改值部分(若存在)则成为两个对象共有，由于shallow copy。所以用spread operator改的地方要明确，其他…，如下图，更改city&name)

Text

Description automatically generated

1. Immer Library, produce method takes original obj and a modification callback. Draftbook monitors all changes and return a new obj

Text

Description automatically generated

**Redux(no data mutation):** state mgmt library for JS apps: keep different parts of UI in sync,due to change of manual/network request/background task.

1. All front end app state is stored in a central repository: JS obj – **Store == State**
2. Makes data flow transparent and predictable
3. State is an array of objects of slices

**Reducer/Event Handler**(1 for each slice(cart/user/products等对象)): 根据action参数，reducer改变store中相应数据(state中)，以immutate 的方式，recuder is pure function。

Text

Description automatically generated

**Action/Event**: JS Obj that describes what just happened,记录状态改变如login/add item..

Redux设计步骤：Graphical user interface, text

Description automatically generated

Dispatch store method can add/remove an action(obj), dispatcher is a method comes with store that calls the recucer. 备注为dispatcher实际逻辑

Text

Description automatically generatedText

Description automatically generated

Subscribe store method triggered from UI layer to fresh UI

**NGRX:**

Npm install –save @ngrx/store

Npm install –save bootstrap

Ngrx package will call reducer ( state, action)

1.set initial state: js obj(not ncessary, recommended )

1. Graphical user interface, diagram

   Description automatically generated

Docker, Containerization, Spring

reactive applications in Angular