ES6高阶

上节课内容总结

- 基于 defineProperty 的数据劫持
- Vue 数据响应式原理解析
- Vue 双向绑定原理总结

课堂主题

1. 基于 Proxy 的数据代理
2. ES6 模块化

课程内容

基于 Proxy 的数据代理

let proxy = new Proxy(target, handler);

- target 是用Proxy包装的被代理对象（可以是任何类型的对象，包括原生数组，函数，甚至另一个代理）
- handler 是一个对象，其声明了代理target 的一些操作，其属性是当执行一个操作时定义代理的行为的函数。

handler 对象的方法

get(target,key[,receiver])

get 方法用来处理获取数据时的劫持行为

set(target,key,newValue[,receiver])

get 方法用来处理设置数据时的劫持行为

has(target,key)

has 方法用来处理在判断是否有该属性时的劫持行为

return true 存在该属性，false 不存在该属性

apply(target, thisArg, argumentsList)

apply 方法用来代理函数的执行，要求 target 必须是一个函数

construct(target, argumentsList)

construct 方法用于拦截 new 操作符.

defineProperty(target, property, descriptor)

defineProperty 方法用于拦截 defineProperty 操作

return Object.defineProperty

deleteProperty(target, property)

deleteProperty 用于拦截对象属性的删除操作

getOwnPropertyDescriptor(obj, key)

getOwnPropertyDescriptor 方法用于拦截 getOwnPropertyDescriptor 操作 getOwnPropertyDescriptor 必须返回一个 object 或 undefined

Object.getOwnPropertyDescriptor() 方法返回指定对象上一个自有属性对应的属性描述符

getPrototypeOf(target)

getPrototypeOf 用于拦截对象调用 getPrototypeOf 方法

Object.getPrototypeOf 查找对象的原型方法

isExtensible(target)

isExtensible 用于拦截对象的isExtensible方法

不可扩展对象

```
- Object.preventExtensions(obj) 阻止对象扩展
- Object.isExtensible() 判断对象是否可以扩展
```

ownKeys(target)

ownKeys 会拦截一下操作： Object.keys()

setPrototypeOf(target,prototype)

setPrototypeOf 方法主要用来拦截 Object.setPrototypeOf().

ES6 模块化

CMD规范 和 AMD 规范

下节课内容

node.JS

• npm包的注册与发布 • yarn、cnpm、nvm工具介绍 • fs加载模板 、 stream方式加载模板 • nodejs中路由介绍 • 新闻列表案例的nodejs实现