Vue3Project DAY01

Vue 数据监听的原理

Vue的核心功能就是当被vue管理的数据有了变化后，将会及时的更新UI。到底如何实现修改变量或对象的属性后，UI可以及时更新?

1. vue2中使用`Object.defineProperties()`监听对象属性的变化。
2. vue3中使用`new Proxy()`为目标对象请一个代理对象，一个代理对象可以管理一整个对象的属性，减少了属性监听器的创建，优化新能。

vue2数据监听的简单实现

vue2中使用`Object.defineProperties()`监听对象属性的变化。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>vuedemo.html</title>
</head>
<body>
  <h1>Vue2数据监听</h1>
  <h2 id="h1">1</h2>
  <button id="btn1">点我后，数字++</button>
  <script>
    var data = {
      num: 1
    }
    // 为data定义属性监听，一旦data对象中属性有变化就更新UI
    Object.defineProperties(data, {
      _num: {value:1, writable:true},
      // 对data的num属性进行监听
      num: {
        // 当外部访问data的num属性时，就会执行该get方法
        get(){
          return this._num
        },
        // 当外部修改data的num属性时，就会执行该set方法，并传入参数
        set(value){
          this._num = value
          // 修改掉num值的同时，此处还可以顺便更新UI
          h1.innerHTML = value
        }
      }
    })
```

```
    btn1.onclick = function(){
      data.num++
    }

  </script>
</body>
</html>
```

vue3的代理模式的简单实现

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>vuedemo.html</title>
</head>
<body>
  <h1>Vue2数据监听</h1>
  <h2 id="h1">1</h2>
  <button id="btn1">点我后，数字++</button>
  <script>
    var data = {
      num: 1
    }
    // 为data定义属性监听，一旦data对象中属性有变化就更新UI
    Object.defineProperties(data, {
      _num: {value:1, writable:true},
      // 对data的num属性进行监听
      num: {
        // 当外部访问data的num属性时，就会执行该get方法
        get(){
          return this._num
        },
        // 当外部修改data的num属性时，就会执行该set方法，并传入参数
        set(value){
          this._num = value
          // 修改掉num值的同时，此处还可以顺便更新UI
          h1.innerHTML = value
        }
      }
    })

    btn1.onclick = function(){
      data.num++
    }
  </script>

  <hr>
  <hr>
  <hr>
```

```
    <h1>Vue3的代理模式</h1>
    <h3 id="h3">1</h3>
    <button id="btn3">点我，数字++</button>
    <script>
      var v3data = {
        count: 1
      }
      // 为v3data对象请一个代理对象，来管理属性的访问及更新
      var dataProxy = new Proxy(v3data, {
        // 当希望获取v3data的某一个属性时，将会执行get
        // obj: 目标对象     key: 要访问的属性名
        get(obj, key){
          return obj[key]
        },
        // 当希望修改v3data的某个属性时，将会执行set
        // obj:目标对象  key:要修改的属性名    value:要修改的属性值
        set(obj, key, value){
          obj[key] = value
          // 更新属性的同时，还可以顺便更新UI
          h3.innerHTML = value
        },
      })

      btn3.onclick = function(){
        dataProxy.count++
      }
      </script>
  </body>
  </html>
```

Vue3 使用代理完成数据监听的常用方法

1. ref() 通常情况下，用于代理基本数据类型与数组
2. reactive() 通常情况下，用于代理js对象

Vue3中的计算属性

vue2中计算属性的写法：

```
  <span>总价格：{{total}}</span>
```

```
data(){
    return {
        price: 10,
        count: 2
    }
}
computed: {
    total(){
        return this.price * this.count
    }
}
```

vue3的计算属性

```
// 声明一个计算属性total，计算总价格
let total = computed(function(){
    // 此处进行数据运算
    return pro.price * pro.count
})
return {total}
```

Vue3中的监听器 watch

vue2的监听器的写法：

```
data(){
    return {
        count: 2
    }
}
watch: {
    count(newValue, oldValue){
        console.log(xxxx)
    }
}
```

vue3的写法：

```
const c = ref(0)
watch(c, (newValue, oldValue)=>{
    console.log('oldValue:', oldValue)
    console.log('newValue:', newValue)
})

// 声明一个属性监听器，监听pro.count的变化
const pro = reactive({count: 0})
watch(()=>pro.count, (newValue, oldValue)=>{
    console.log('oldValue:', oldValue)
    console.log('newValue:', newValue)
})
```

Vue3.2提供的setup语法糖

在了解了vue3中动态数据绑定的语法、方法声明的语法、计算属性的语法、监听器的语法后，发现几乎所有的代码都在setup方法内部编写，并且template中需要的资源都需要通过return返回。vue3.2提供了setup语法糖来简化这些语法：

```
<script lang="ts" setup>
    // 使用了setup语法糖后，在此处编写的代码，相当于在setup方法中编写
    // 在此处声明的变量都将会自动导出，template中可以直接使用
    import { ref, reactive, computed, watch } from 'vue';
    const aaa=4;
    const bbb=10;
    const count = ref(10)
    // 直接声明minus方法
    function minus(){
        // ref.value可以访问到代理的目标对象
        count.value--
    }

    // 准备一件商品
    const pro = reactive({
        name: '钢管',
        price: 68,
        count: 3
    })
    function jiajia(){
        // pro.value.count++
        pro.count++
    }
    function jianjian(){
        pro.count--
    }
    function changePro(){  // 换一件商品
        pro.name = '榴莲'
        pro.price = 98
        pro.count = 10
        // pro = reactive({
```

```
        //   name: '榴莲',
        //   price: 98,
        //   count: 10
        // })
    }

    // 声明一个计算属性total，计算总价格
    let total = computed(function(){
        // 此处进行数据运算
        return pro.price * pro.count
    })

    // 声明一个属性监听器，监听pro.count的变化
    watch(()=>pro.count, (newValue, oldValue)=>{
        console.log('oldValue:', oldValue)
        console.log('newValue:', newValue)
    })

</script>
```

组合式API致力于将同一个业务逻辑所需要用到的变量、方法写在同一个位置，并不是像vue2选项式API一样，写写data，写写methods，写写mounted（跳着写），更有利于后续代码维护。

Vue3中使用axios发送请求

1. 安装axios。

   ```
   npm i axios -S   # (大写S)
   ```

2. 需要调用axios的方法，需要引入`MyAxios.ts`。

   ```
   import myaxios from '@/http/MyAxios.ts'
   ```

3. 发送http请求，加载演员列表。

   ```
   https://web.codeboy.com/bmdapi/movie-actors?page=1&pagesize=100
   ```

案例：访问：`/actors`看到`views/Actors.vue`。

百慕大影城前台移动端项目实践

该项目供普通用户使用，提供了查询不同类别的电影列表，查看电影详情，查询电影院、查询放映厅、选座、下订单等功能。

技术选型： Vue3 VueRouter Vuex Typescript Vant组件库等。

项目初始化

1. 新建一个脚手架项目。`bmdstudios-mobile-client`

```
# 找一个干净地方： demo01/
vue  create  bmdstudios-mobile-client
# 依次选择
Manually select features
# 选择以下6项 （注意选中Typescript）
 (*) Babel
 (*) TypeScript
 (*) Router
 (*) Vuex
 (*) CSS Pre-processors
 (*) Linter / Formatter
# 选择3.x
3.x
# 一路回车
```

2. 安装模块：

```
cd bmdstudios-mobile-client
npm  i  axios  -S
# 安装vant
```

3. 启动脚手架web服务：

```
npm run serve
```

在项目中引入vant组件库

1. 安装所需模块：

```
npm i vant
npm i unplugin-vue-components -D
```

2. 配置插件：如果是基于 `vue-cli` 的项目，在 `vue.config.js` 文件中配置插件：

```
const { defineConfig } = require('@vue/cli-service')
const { VantResolver } = require('unplugin-vue-components/resolvers');
const ComponentsPlugin = require('unplugin-vue-components/webpack');

module.exports = defineConfig({
  transpileDependencies: true,
  configureWebpack: {
    plugins: [
```

```
      ComponentsPlugin({
        resolvers: [VantResolver()],
      }),
    ],
  },
})
```

3. 使用组件：

```
<template>
  <van-button type="primary" />
</template>
```

搭建项目的初始布局结构

整个页面包含两部分：

1. main部分
2. 底部选项卡部分

基于嵌套路由的方式来动态改变页面上半部分 （main部分） 的内容。

在`HomeView.vue`中添加一个底部选项卡。