网址https://vuejs.org/

安装

npm install -g @Vue/cli

查看安装成功

vue --version

创建

指令：vue create vue-demo

空格选这些

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)
 (*) Babel
 ( ) TypeScript
>(*) Progressive Web App (PWA) Support
 ( ) Router
 ( ) Vuex
 ( ) CSS Pre-processors
 ( ) Linter / Formatter
 ( ) Unit Testing
 ( ) E2E Testing
```

然后选vue3

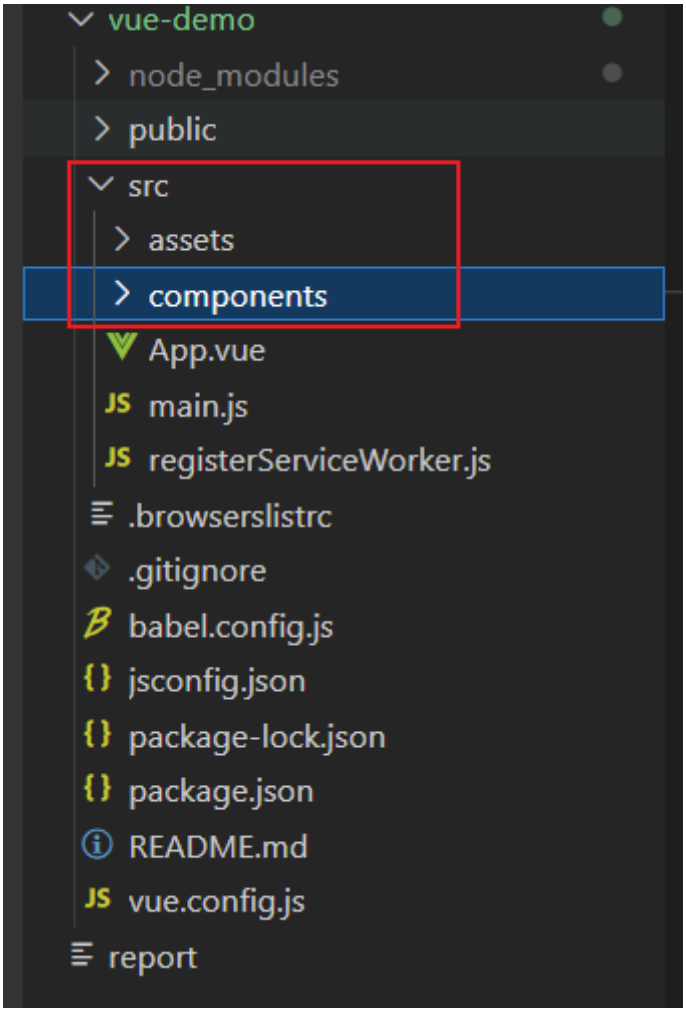
进入项目和运行

cd vue-demo

npm run serve

项目结构

src是重点，assets是静态资源，components是组件。其他是配置文件



高亮插件

vetur：vue2

volar：vue3

App.vue

template是写html的

script是写js的

总结

vue就是实现动态渲染

{{}}

双大括号会解释成普通文本，而不是html

动态显示数据，data()是函数

也支持js表达式,注意是表达式

{{ message +10 }}

{{ ok? 'yes': 'no'}}

```
demo > src > App.vue > {} script > default > data > message
<template>
  <div class="hello">
      <h3>模板学习</h3>
      <p>{{ message }}</p>
  </div>
</template>

<script>
export default (await import('vue')).defineComponent({
  name: 'helloworid',
  data(){
    return{
      message: "测试"
    }
  }
})
</script>
```

v-html

识别html文本

```
demo > src > App.vue > {} template > div.hello > div
1 <template>
2   <div class="hello">
3       <h3>模板学习</h3>
4       <p>{{ message }}</p>
5       <div>{{a }}</div>
6       <div v-html="a"></div>
7   </div>
8 </template>
9
0 <script>
1 export default (await import('vue')).defineComponent({
2   name: 'helloworid',
3   data(){
4     return{
5       message: "测试",
6       a: "<a href='https://www.baidu.com/'>百战</a>"
7     }
8   }
9 })
0 </script>
```

v-bind

v-bind可简写成:

也就是v-bind:id写成:id

属性动态变化，这里例如id

```vue
<template>
  <div class="hello">
    <h3>模板学习</h3>
    <p>{{ message }}</p>
    <div>{{a }}</div>
    <div v-html="a"></div>
    <div v-bind:id="c">zyj</div>
  </div>
</template>

<script>
export default (await import('vue')).defineComponent({
  name: 'helloworid',
  data(){
    return{
      message: "测试",
      a: "<a href='https://www.baidu.com/'>百战</a>",
      c:1001
    }
  }
})
```

v-if和v-else

只有当是true才会渲染

```vue
<template>
  <div class="hello">
  <p v-if="flag">我是庄淯津</p>
  <p v-else></p>
  </div>
</template>

<script>
export default (await import('vue')).defineComponent({
  name: 'helloworid',
  data(){
    return{
      flag:true
    }
  }
})
</script>
```

v-show

还有

，他是单独的，不是if-lese

if和show的区别

if是真正的条件渲染，添加代码或者删除代码。 切换开销高。

而show是根据css显示和不显示。 频繁使用则用show。

v-for

v-for="item in items" ，数组items跟单个数item遍历

如果你修改newList下标i的数据，渲染是直接渲染下标i，不是重头遍历

为了给Vue提示追踪节点身份，需要用到唯一的key，可以提高性能。

- {{ i.title }}

```
ue-demo > src > App.vue > {} template > div.hello > ul > li
1   <template>
2     <div class="hello">
3         <h3>列表渲染</h3>
4         <ul>
5           <li v-for="i in newList">{{ i.title }}</li>
6         </ul>
7     </div>
8   </template>
9
10  <script>
11  export default (await import('vue')).defineComponent({
12    name: 'helloworid',
13    data(){
14      return{
15        newList:[
16          {
17            id:1,
18            title:"1"
19          },
20          {
21            id:2,
22            title:"2"
23          }
24        ]
25      }
26    }
27  })
28  </script>
```

v-on

可缩写成@符号：<button @click="counter +=1">点击:counter{{ counter }}

监听事件

```
-demo > src > App.vue > {} template > div.hello > button
1   <template>
2     <div class="hello">
3         <button v-on:click="counter +=1">点击:counter{{ counter }}</button>
4     </div>
5   </template>
6
7   <script>
8   export default (await import('vue')).defineComponent({
9     name: 'helloworid',
0     data(){
1       return{
2           counter:1
3       }
4     }
5   })
6   </script>
7
8
```

事件也可以触发方法

```
mo > src > App.vue > {} script > default > data
<template>
  <div class="hello">
      <button @click="counter +=1">点击:counter{{ counter }}</button>
      <button @click="clickhand">点击:counter{{ counter }}</button>
  </div>
</template>

<script>
export default (await import('vue')).defineComponent({
  name: 'helloworid'
  data(){
    return{
        counter:1
    }
  },
  methods:{
    clickhand(){
      console.log("哈哈哈")
    }
  }
})
</script>
```

methods中读取data中属性

在事件中读取data的值需要this

```
<template>
  <div class="hello">
      <button @click="counter +=1">点击:counter{{ counter }}</button>
      <button @click="clickhand">点击:counter{{ counter }}</button>
  </div>
</template>

<script>
export default (await import('vue')).defineComponent({
  name: 'helloworid',
  data(){
    return{
        counter:1
    }
  },
  methods:{
    clickhand(){
      this.counter="点击了"
    }
  }
})
</script>
```

event

```
<script>
export default (await import('vue')).defineComponent({
  name: 'helloworid',
  data(){
    return{
        counter:1
    }
  },
  methods:{
    //事件中的event是原生DOM event，会自动传入一个对象
    clickhand(evenet){
      evenet.target.innerHTML = "666"
    }
  }
})
</script>
```

事件传递参数

```
<template>
  <div class="hello">
      <button @click="say('hi')">hi</button>
      <button @click="say('what')">what</button>
  </div>
</template>

<script>
export default (await import('vue')).defineComponent({
  name: 'helloworid',
  data(){
    return{
        counter:1
    }
  },
  methods:{
    say(data)
    {
      console.log(data)
    }
  }
})
</script>
```

v-model

双向数据绑定，输入就获取到

```
<input type="text" v-model="username">
<p>{{ username }}</p>
```

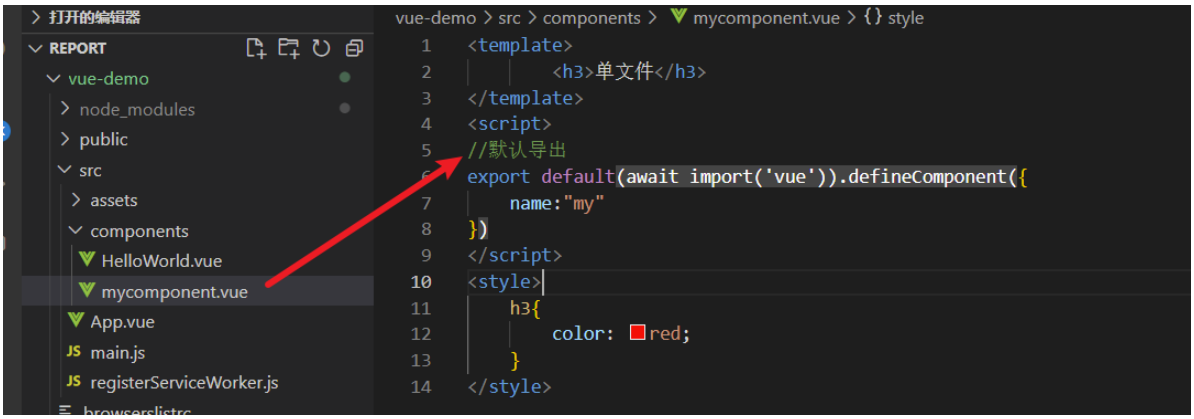
修饰符lazy

只有当回车，失去焦点才显示

修饰符trim

去掉空格

组件开发



App.vue是根组件

```
<template>
    <mycomponent/>
    <div class="hello">
        <input type="text" v-model="username">
        <p>{{ username }}</p>
        <button @click="clickname">获取数据</button>
    </div>
</template>

<script>
//导入组件
import mycomponent from "./components/mycomponent.vue"

export default(await import('vue')).defineComponent({
    name:'app',
    components:{
        mycomponent
    }
})
</script>

<style>
</style>
```

scope

如果加了scoped就只在当前组件中生效

```
<style scoped>
</style>
```

组件交互

将App.Vue中title传给组件mycomponent的a

```
<template>
  <mycomponent :a="title"/>
  <div class="hello">
      <input type="text" v-model="username">
      <p>{{ message }}</p>
      <button @click="clickname">获取数据</button>
  </div>
</template>

<script>
//导入组件
import mycomponent from "./components/mycomponent.vue"

export default(await import('vue')).defineComponent({
  name:'app',
  components:{
    mycomponent
  },
  data()
  {
    return{
      title:"标题"
    }
  }

})
</script>
```

props父组件传递到子组件

接着mycomponent通过props获取a

report　App.vue　mycomponent.vue

vue-demo > src > components > mycomponent.vue > {} script > default > props > a

```vue
<template>
        <h3>单文件{{ a }}</h3>
</template>
<script>
//默认导出
export default(await import('vue')).defineComponent({
    name:"my",
    props:{
        a:{
            type:String,
            //默认值
            default:"default"
        }
    }
  })
</script>
<style>
    h3{
        color: red;
    }
</style>
```

数组

```
props:{
    a:{
        type:String,
        //默认值
        default:"default"
    },
    b:{
        type:Array,
        //数组和对象必须使用函数进行返回
        default:function()
        {
            return []
        }
    }
}
```

子组件数据传递给父组件

子组件创建send事件，send事件自定义事件获取值，父组件通过自定义事件拿到值

mycomponent.vue ✕ App.vue

vue-demo > src > components > mycomponent.vue > {} script > default > methods

```vue
<template>
        <h3>单文件</h3>
        <button @click="send">儿子点击传递</button>
</template>
<script>
//默认导出
export default(await import('vue')).defineComponent({
    name:"my",
    data()
    {
        return{
            son:"我是儿子传给爸爸"
        }
    },
    methods:{
    //儿子发送给爸爸的数据
    send()
    {
        //自定义事件获取son值
      this.$emit("onEvent",this.son)
    }
  }
  })
</script>
<style>
```

```vue
<template>
  <mycomponent @onEvent="gethand" />
  <div class="hello">

  </div>
</template>

<script>
//导入组件
import mycomponent from "./components/mycomponent.vue"

export default(await import('vue')).defineComponent({
  name:'app',
  components:{
    mycomponent
  },
  methods:{
    gethand(data)
    {
        console.log(data)
    }
  }
})
</script>
```

vue的生命周期

为了方便记忆，我们可以将他们分类:

前，后

创建时: beforeCreate 、 created
渲染时: beforeMount 、 mounted （例如渲染后mounted就发送网络请求）
更新时:beforeUpdate updated
卸载时:beforeUnmount 、unmounted （卸载后停掉定时器之类的东西 ）

例如生命周期函数

```
<script>
//导入组件
import mycomponent from "./components/mycomponent.vue"

export default(await import('vue')).defineComponent({
  name:'app',
  components:{
    mycomponent
  },
  methods:{
    gethand(data)
    {
        console.log(data)
    }
  },
  beforeCreate()
  {

  }
})
```

vue安装第三方

npm install --save swiper@8.1.6

```
<script>
//引入第三方
import {Swiper, SwiperSlide} from 'swiper/vue';
import 'swiper/css';
export default(await import('vue')).defineComponent({
  name:'app',
  components:{
    Swiper,SwiperSlide
  }
})
</script>

<style scoped>
</style>
```

axios

安装

npm install --save axios

组件引用

//引入第三方

import axios from "axios"

渲染完后使用

get

```
mounted()
{
  axios({
    method:"get",
    url:"https://lenovo.ilive.cn/?f=stee"
  }).then(res =>{
    console.log(res.data);
  })
}
```

post

要注意转换

npm install --sava querystring

import querystring from "querystring"

因为是对象类型，不是string类型，所以需要转换

```
axios({
  method:"post",
  url:"http://iwenwiki.com/api/blueberrypai/login.php",
  data:querystring.stringify({
    user_id:"iwen@qq.com",
    password:"iwen123",
    verification_code:"crfvw"
  })
}).then(res=>{
  console.log(res.data)
})
```

快捷

```
axios.get("http://iwenwiki.com/api/blueberrypai/getChengpinDetails.php").then(res =>{
  console.log(res.data)
})
```

```
axios.post("http://iwenwiki.com/api/blueberrypai/login.php",querystring.stringify({
    user_id:"iwen@qq.com",
    password:"iwen123",
    verification_code:"crfvw"})).then(res =>{
  console.log(res.data)
})
```

全局引用，注意$axios

在main.js中引入后挂载

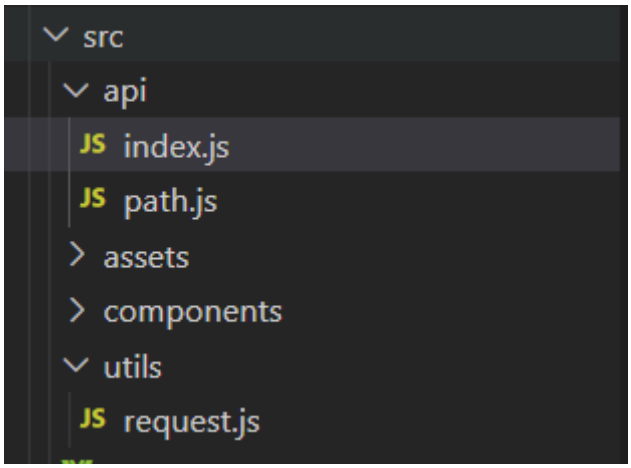
report App.vue main.js

vue-demo > src > main.js > ...

```
import { createApp } from 'vue'
import App from './App.vue'
import './registerServiceWorker'

import axios from "axios"
const app= createApp(App)
app.config.globalProperties.$axios = axios
app.mount('#app')
```

```
this.$axios.get("http://iwenwiki.com/api/blueberrypai/getChengpinDetails.php").then(res =>{
  console.log(res.data)
})
this.$axios.post("http://iwenwiki.com/api/blueberrypai/login.php",querystring.stringify({
    user_id:"iwen@qq.com",
    password:"iwen123",
    verification_code:"crfvw"})).then(res =>{
  console.log(res.data)
})
```

网络请求封装



创建request.js封装请求

```
import axios from "axios";
import querystring from "querystring"

const instance = axios.create({
    //网络请求的公共配置
    timeout:5000
})
const errorHandle =(status,info) =>{
    switch(status){
    case 400:
    console.log("语义有误");
    break;
    case 401:
```

```
    console.log("服务器认证失败");
    break;
    case 403:
    console.log("服务器拒绝访问");
    break;
    case 404:
    console.log("地址错误");
    break;
    case 500:
    console.log("服务器遇到意外");
    break;
    case 502:
    console.log("服务器无响应");
    break;
    default:
    26
    console.log(info);
    break;
    }
}
//拦截器最常用

//发送数据之前
instance.interceptors.request.use(
    config =>{
        //post要string
        if(config.method =="post"){
            config.data = querystring.stringif(config.data)
        }
        //config包含网络请求的所有信息
        return config;
    },
    error =>{
        return Promise.reject(error)
    }
)
instance.interceptors.response.use(
    response =>{
        return response.status == 200? Promise.resolve(response)
:Promise.reject(response)
    },
    error =>{
        const {response} = error;
        errorHandle(response.status,response.info)
    }
)

//导出
export default instance;
```

创建path.js封装路径

```
const base = {
    baseUrl:"http:iwenwiki.com/",
    chengpin:"/api/blueberrypai/getChengpinDetails.php"
}
//导出
export default base;
```

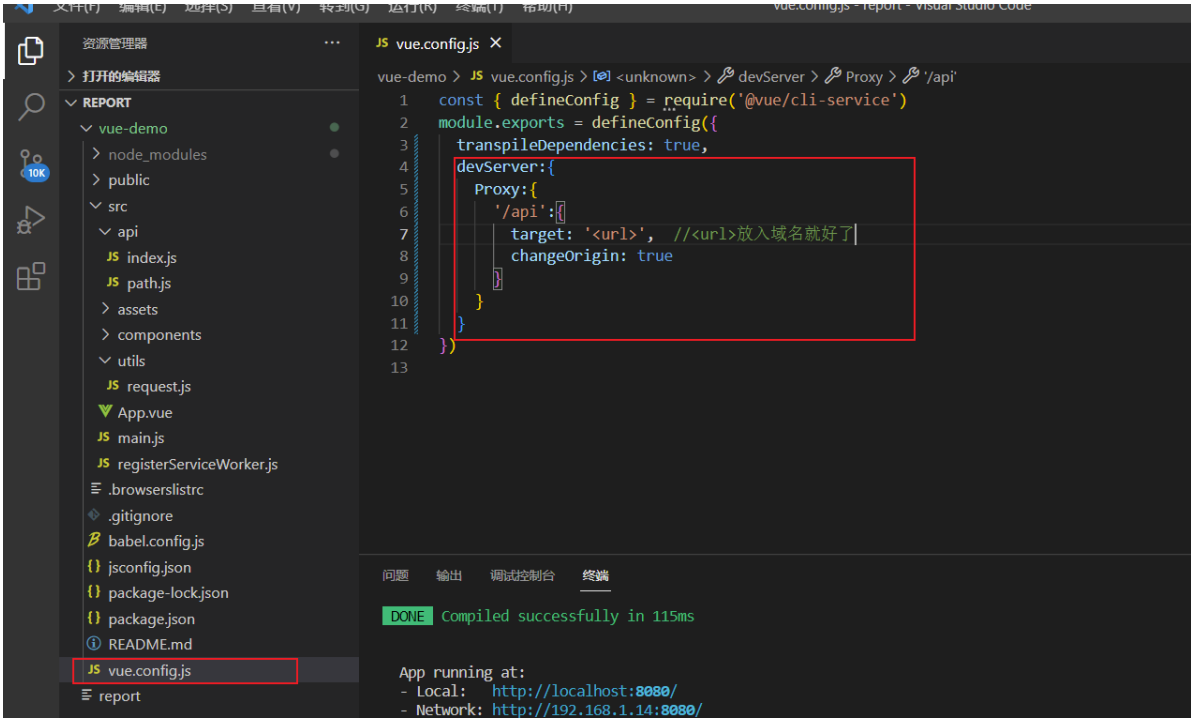
创建index.js写好请求

```
import axios   from "../utils/request"
import path from "./path"

const api ={
    getChengpin()
    {
        return axios.get(path.baseUrl+path.chengpin)
    }
}
export default api
```

App.vue

```
import api from "../src/api/index"
mounted()
  {
    api.getChengpin().then(res =>{
      console.log(res.data);
    })
  }
```

跨域问题



注意url不用写那么长了

```
axios.get("/api/FingerUnion/list.php").then(res =>{
     console.console.log(res.data);
   })
```

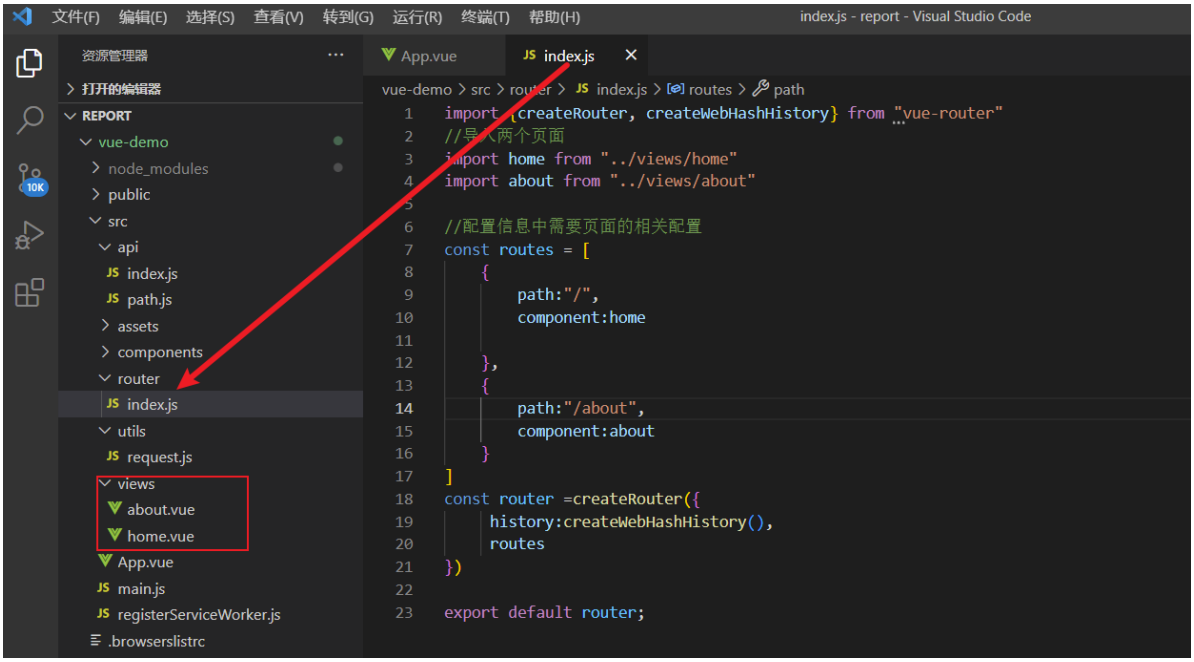
重启服务器

路由配置

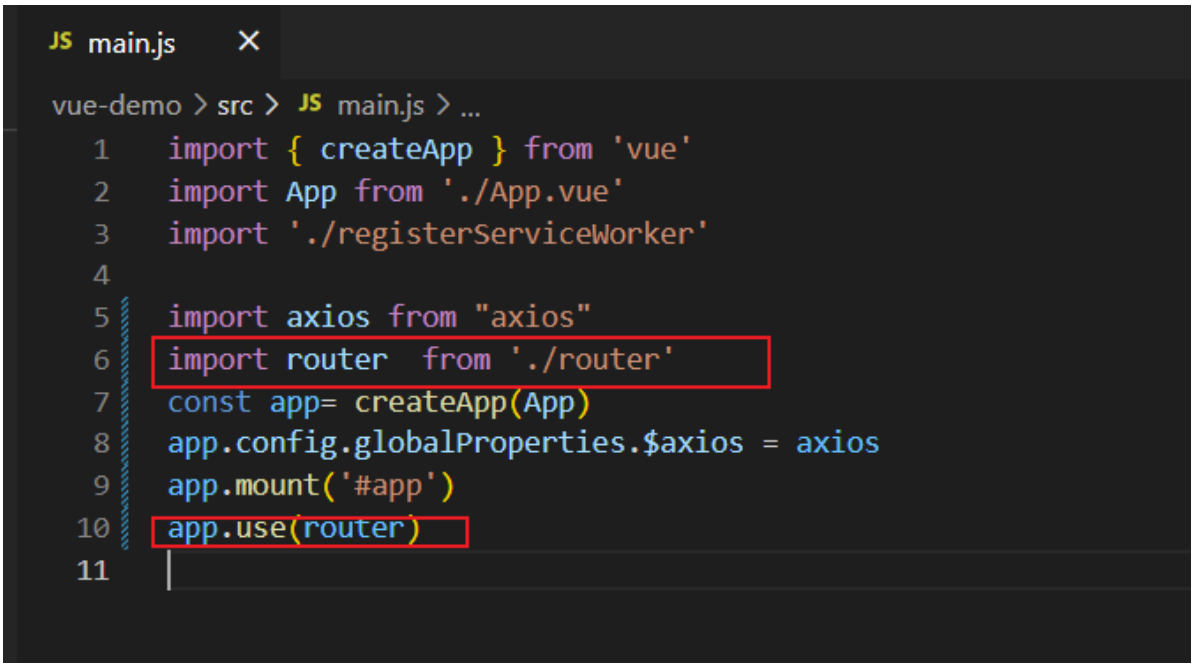
安装路由

npm install --save vue-router

创建vue然后写路由



main.js引入



或者

```
import router  from './router'

createApp(App).use(router).mount('#app')
```

App.Vue显示

```
查看(V)  转到(G)  运行(R)  终端(T)  帮助(H)                App.vue - r
App.vue ×
vue-demo > src > App.vue > {} template
1  <template>
2    <!-- 路由的显示入口 -->
3    <router-view></router-view>
4    <div>
5
6    </div>
7  </template>
```

跳转

```
App.vue ×
vue-demo > src > App.vue > {} template
1  <template>
2    <!-- 路由的显示入口 -->
3    <router-view></router-view>
4    <!-- 路由的跳转 -->
5    <router-link to="/">首页</router-link>
6    <router-link to="/about">关于</router-link>
7    <div>
8
9    </div>
10 </template>
11
12 <script>
13 import axios from "axios"
14 export default(await import('vue')).defineComponent({
15
16 })
17 </script>
18
19 <style scoped>
20 </style>
21
22
```

路由传递参数

异步加载

```
{
    path:"/new/:name",
    component:() => import("../views/new.vue")
}
```

路径设参

```
{
    path:"/new/:name",
    component:() =>{"../views/new.vue"}
}
```

传参

App.vue

vue-demo > src > App.vue > {} template > div

```
<template>
  <!-- 路由的显示入口 -->
  <router-view></router-view>
  <!-- 路由的跳转 -->
  <router-link to="/">首页</router-link>
  <router-link to="/about">关于</router-link>
  <router-link to="/new/参数">新来的</router-link>
  <div>

```

渲染

new.vue

vue-demo > src > views > new.vue > {} template > p

```
<template>
    <h3>new</h3>
    <p>{{ $route.params.name }}</p>
</template>
```

嵌套路由

```
{
    path:"/new/:name",
    component:() => import("../views/new.vue"),
    children:[
        {
            //二级导航不要加/
            path:"children",
            component:() =>import("../views/children.vue")
        }
    ]
}
```

```
e-demo > src > views > new.vue > {} template > router-link
1   <template>
2       <h3>new</h3>
3         <!-- 路由的显示入口 -->
4       <router-view></router-view>
5       <router-link to="/new/children">childern</router-link>
6       <p>{{ $route.params.name }}</p>
7   </template>
```

默认重定向

```
{
    path:"/new/:name",
    component:() => import("../views/new.vue"),
    //默认重定向
    redirect:"/new/children",
    children:[
        {
            //二级导航不要加/
            path:"children",
            component:() =>import("../views/children.vue")
        }
    ]
}
```

vue状态管理

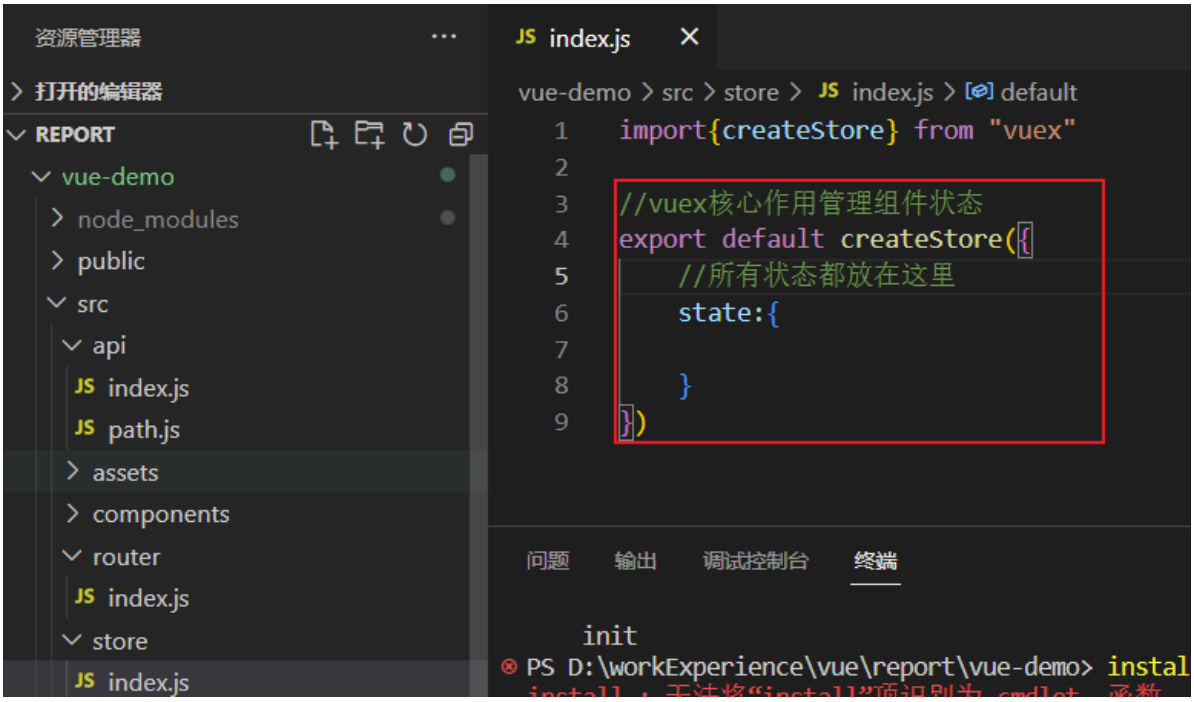
组件之间传递，关系太过于复杂。

所以加一层，提供一个集中的管理方案，也就是中介。

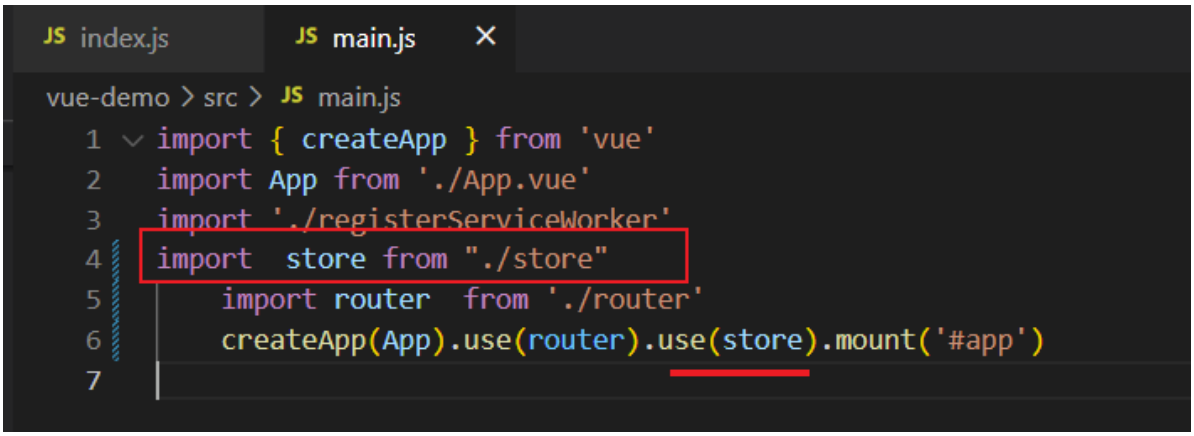
下载

npm install vuex --save

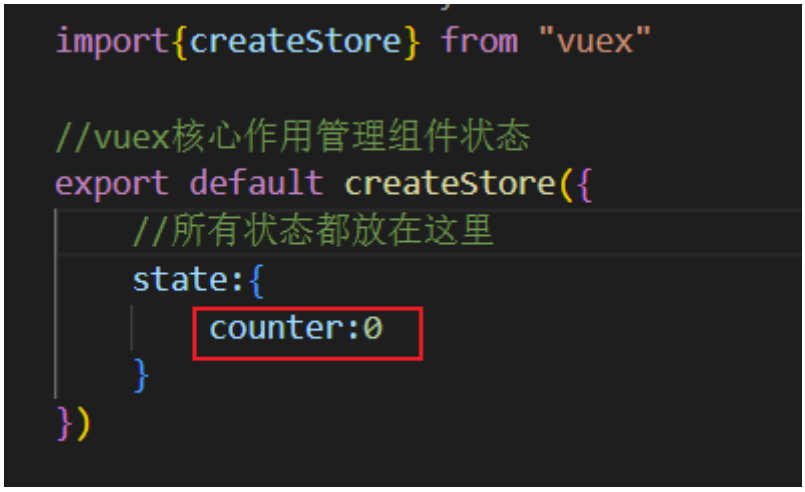
创建



main.js引入



共享参数



```
vue-demo > src > App.vue > {} template > div > p
<template>
  <!-- 路由的显示入口 -->
  <router-view></router-view>
  <!-- 路由的跳转 -->
  <router-link to="/">首页</router-link>
  <router-link to="/about">关于</router-link>
  <router-link to="/new/参数">新来的</router-link>
  <div>
    <p>{{$store.state.counter }}</p>
  </div>
</template>

<script>
import axios from "axios"
```

或者另一种方式

```
App.vue
vue-demo > src > App.vue > {} template > div > p
<template>
  <div>
    <p>{{counter }}</p>
  </div>
</template>

<script>
import {mapState} from "vuex"
export default(await import('vue')).defineComponent({
    computed:{
      ...mapState(["counter"])
    }
})
</script>

<style scoped>
</style>
```

条件获取

getters获取值

```
//vuex核心作用管理组件状态
export default createStore({
    //所有状态都放在这里
    state:{
        counter:0
    },
    getters:{
        getCount(state)
        {
            return state.counter>0? state.counter:"不符合要求"
        }
    }
})
```

```
App.vue   index.js
vue-demo > src > App.vue > {} template > div
1  <template>
2    <div>
3      <p>{{$store.getters.getCount  }}</p>
4    </div>
5  </template>
6
```

或者直接

```
index.js   App.vue
vue-demo > src > App.vue > {} script > default > computed > <unknown>
1  <template>
2    <div>
3      <p>{{getCounter  }}</p>
4    </div>
5  </template>
6
7  <script>
8  import { mapGetters} from "vuex"
9  export default(await import('vue')).defineComponent({
10     computed:{
11       ...mapGetters(["getCounter"])
12     }
13 })
14 </script>
15
16 <style scoped>
17 </style>
18
19
```

mutations修改值

```
//vuex核心作用管理组件状态
export default createStore({
    //所有状态都放在这里
    state:{
        counter:0
    },
    getters:{
        getCounter(state)
        {
            return state.counter>0? state.counter:"不符合要求"
        }
    },
    mutations:{
        addCounter(state)
        {
            state.counter++
        }
    }
})
```

```
<template>
  <div>
    <p>{{getCounter  }}</p>
    <button @click="addClick">增加</button>
  </div>
</template>

<script>
import { mapGetters} from "vuex"
export default(await import('vue')).defineComponent({
    computed:{
      ...mapGetters(["getCounter"])
    },
    methods:{
        addClick(){
          //固定调用方式
            this.$store.commit("addCounter")
        }
    }
})
</script>

<style scoped>
</style>
```

可以传参

```
mutations:{
    addCounter(state,number)
    {
        state.counter+=number
    }
}
```

```
methods:{
    addClick(){
     //固定调用方式
       this.$store.commit("addCounter",15)
    }
}
```

或者直接

```
<script>
import { mapGetters,mapMutations } from "vuex"
export default(await import('vue')).defineComponent({
    computed:{
      ...mapGetters(["getCounter"])
    },
    methods:{
      ...mapMutations(["addCounter"]),
      addClick(){
       //固定调用方式
         this.addCounter(15)
      }
    }
})
</script>
```

actions网络请求异步

注意import axios from "axios"

addCounter是方法

```
//为异步操作所准备的
actions:{
    asyncAddCounter({commit})
    {
        axios.get("http://iwenwiki.com/api/generator/list.php")
        .then(res =>{
            console.log(res.data)
            commit("addCounter",res.data[0])
        })
    }
}
```

```
methods:{
  ...mapMutations(["addCounter"]),
  addClick(){
   //固定调用方式
     this.addCounter(15)
  },
  addAsync(){
   this.$store.dispatch("asyncAddCounter")
  }
}
```

或者直接

```
<script>
import { mapGetters,mapMutations,mapActions } from "vuex"
export default(await import('vue')).defineComponent({
    computed:{
      ...mapGetters(["getCounter"])
    },
    methods:{
      ...mapMutations(["addCounter"]),
      ...mapActions(["asyncAddCounter"]),
       addClick(){
        //固定调用方式
          this.addCounter(15)
       },
       addAsync(){
        this.asyncAddCounter()
       }
    }
})
</script>
```

ref和react其实就是methods，data啥的写一起了

数据

```
e-demo > src > App.vue > {} script > default > setup > names
<template>
  <div>
    <p>{{ message  }}</p>
    <ul>
      <li v-for="(item,index) in names.list" :key="index">{{ item }}</li>
    </ul>
  </div>
</template>

<script>
import { ref,reactive} from "vue"
export default(await import('vue')).defineComponent({
  setup(){
    const message = ref("消息")

    //对象或者数组
    const names =reactive({ list:["a","b","c"]})
    //需要return
    return {
      message,names
    }
  }

})
</script>
```

事件

注意要通过message.value修改消息

```
    </ul>
    <button @click="click">点击</button>
  </div>
</template>

<script>
import { ref,reactive} from "vue"
export default(await import('vue')).defineComponent({
  setup(){
    const message = ref("消息")

    //对象或者数组
    const names =reactive({ list:["a","b","c"]})

    //方法
    function click(){
      message.value ="我是新消息"
    }
    //需要return
    return {
      message,names,click
    }
  }

})
</script>
```

props

忽略

生命周期函数

不需要return

```
<script>
import { ref,reactive, onMounted} from "vue"
export default(await import('vue')).defineComponent({

  setup(){
    const message = ref("消息")

    //可以有多个
    onMounted(()=>{
      message.value="生命周期1"
    })
    onMounted(()=>{
      message.value="生命周期2"
    })
    //需要return
    return {
      message
    }
  }

})
</script>
```

父组件给子组件传数据

忽略

elementUI

网址

https://element-plus.org/zh-CN/#/zh-CN

安装/注意2和3有区别

```
npm install element-plus --save
```

main.js引入

```
JS main.js M ×
vue-demo > src > JS main.js
1  import { createApp } from 'vue'
2  import App from './App.vue'
3  import './registerServiceWorker'
4  // import  store from "./store"
5  //      import router  from './router'
6  import ElementPlus from 'element-plus'
7  import 'element-plus/dist/index.css'
8      createApp(App).use(ElementPlus).mount('#app')
9
```

使用

对应

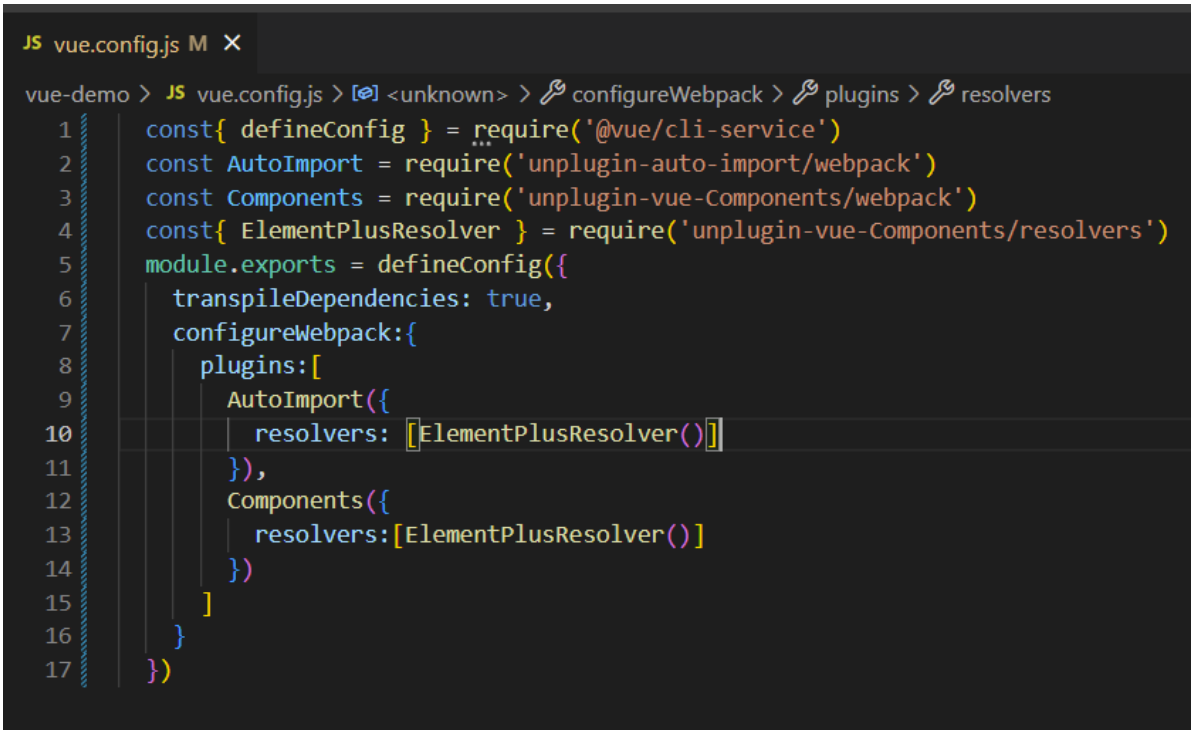
```
<template>
  <div>
    <el-switch v-model="value1" />
    <el-switch
      v-model="value2"
      class="ml-2"
      style="--el-switch-on-color: #13ce66; --el-switch-off-color: #ff4949"
    />
  </div>
</template>

<script>
import { ref } from 'vue'
export default(await import('vue')).defineComponent({
  setup(){
    const value1 = ref(true)
    const value2 = ref(true)
    return{
      value1,value2
    }
  }


})
</script>
```

按需导入插件

安装

npm install -D unplugin-vue-components unplugin-auto-import

修改配置文件vue.config.js



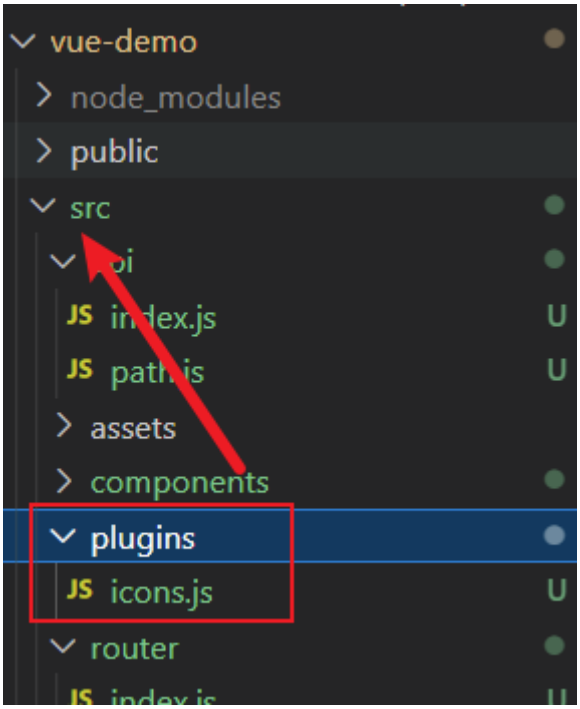

```
const{ defineConfig } = require('@vue/cli-service')
const AutoImport = require('unplugin-auto-import/webpack')
const Components = require('unplugin-vue-Components/webpack')
const{ ElementPlusResolver } = require('unplugin-vue-Components/resolvers')
module.exports = defineConfig({
  transpileDependencies: true,
  configureWebpack:{
    plugins:[
      AutoImport({
        resolvers: [ElementPlusResolver()]
      },
      Components({
        resolvers:[ElementPlusResolver()]
      })
    ]
  }
})
```

然后直接用就好了，不需要main.js引入全局的

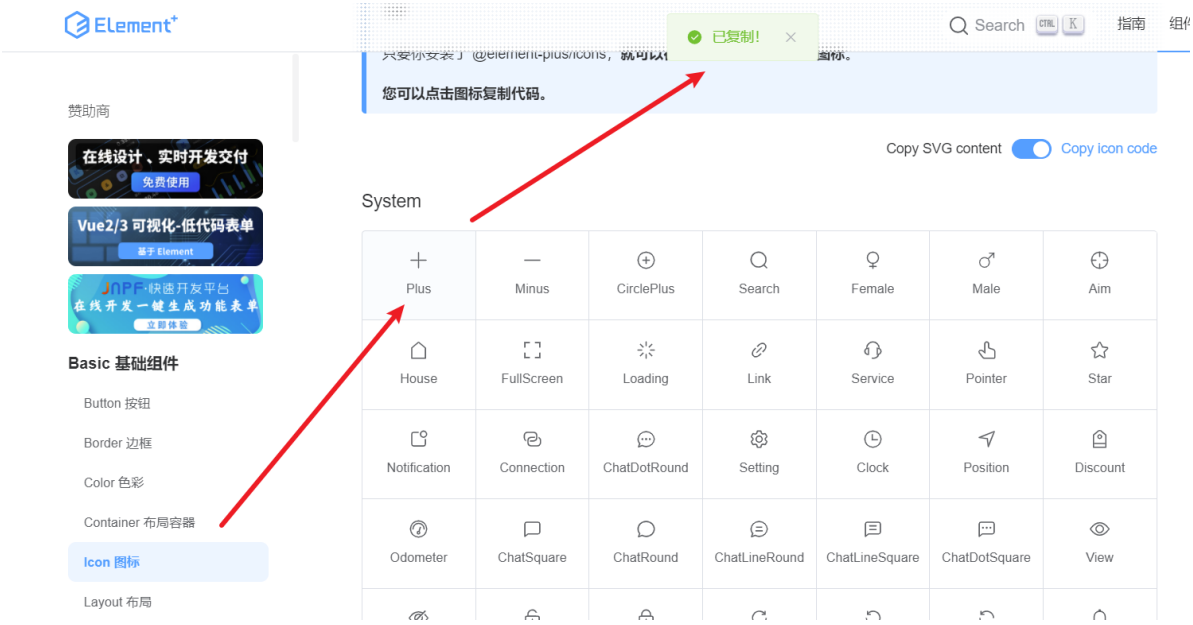
安装图标字体才能用图标

配置

npm install @element-plus/icons-vue




```
import * as components from "@element-plus/icons-vue";
export default {
    install: (app) => {
        for (const key in components) {
            const componentConfig = components[key];
            app.component(componentConfig.name,componentConfig);
        }

    },
};
```




```
<el-icon :size="20" color="red"><Plus /></el-icon>
```

main.js引入

注意./plugins/icons是刚才配置的