# 重构的收获

1. String.fromCharCode（numX） 可接受一个指定的 Unicode 值，然后返回一个字符串。 1

参数：numX ，必填的，一个或多个 Unicode 值。

2. @click="$router.go(-1)" 返回上一个页面

### 3. 移动端点击延迟事件

1. 移动端浏览器在派发点击事件的时候，通常会出现300ms左右的延迟

2. 原因: 移动端的双击会缩放导致click判断延迟

# 解决方法：Fastclick

// 安装 cnpm install fastclick -S

// 引入 import FastClick from 'fastclick'

    // 使用 FastClick.attach(document.body);

使用js 引用：<script type='application/javascript' src='/path/to/fastclick.js'></script>

使用jquery ： $(function() {

        FastClick.attach(document.body);

    });

// 没使用jquery的时候

    if ('addEventListener' in document) {

        document.addEventListener('DOMContentLoaded', function() {

            FastClick.attach(document.body);

        }, false);

    }

# 4.go back与return有什么区别

go back回去,回到某一个地方  
 return是从A地到B地,然后强调了又从B地回到A地  
 而且return还有一个用法就是指借了东西,归还回去

# 5. [Fetch](https://www.cnblogs.com/libin-1/p/6853677.html)

**Fetch获取数据**

假设我们想通过GitHub获取一个仓库，我们可以像下面这样使用：

fetch('https://api.github.com/users/chriscoyier/repos');

Fetch会返回Promise，所以在获取资源后，可以使用.then方法做你想做的。

fetch('https://api.github.com/users/chriscoyier/repos')

.then(response => {/\* do something \*/})

Github返回的响应是JSON格式的，所以调用response.json方法来转换数据。还有其他方法来处理不同类型的响应。如果请求一个XML格式文件，则调用response.text。如果请求图片，使用response.blob方法

# 列如： fetch('https://api.github.com/users/chriscoyier/repos')

# .then(response => response.json())

# .then(data => {

# // data就是我们请求的repos

# console.log(data)

# });

# Fetch发送数据

使用Fetch发送也很简单，只需要配置三个参数。

第一个参数是设置请求方法（如post、put或del），Fetch会自动设置方法为get。

第二个参数是设置头部。因为一般使用JSON数据格式，所以设置ContentType为application/json。

第三个参数是设置包含JSON内容的主体。因为JSON内容是必须的，所以当设置主体时会调用JSON.stringify。

列如： let content = {some: 'content'};

// The actual fetch request

fetch('some-url', {

method: 'post',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(content)

})

// .then()...

**6. async**

它作为一个关键字放到函数前面，用于表示函数是一个异步函数

async 函数返回的是一个promise 对象，如果要获取到promise 返回值，我们应该用then 方法。Promise 有一个resolved，这是async 函数内部的实现原理，如果async 函数中有返回一个值 ,当调用该函数时，内部会调用Promise.solve() 方法把它转化成一个promise 对象作为返回，但如果timeout 函数内部抛出错误呢？ 那么就会调用Promise.reject() 返回一个promise 对象， 这时修改一下timeout 函数

async function timeout(flag) {

if (flag) {

return 'hello world'

} else {

throw 'my god, failure'

}

}

console.log(timeout(true)) // 调用Promise.resolve() 返回promise 对象。

console.log(timeout(false)); // 调用Promise.reject() 返回promise 对象**。**

**7.** XLink，即 XML 链接语言，用于在 XML 文件上创建内部和外部链接，以及与这些链接相关联的元数据。

属性（xlink:href）值（*URL*） 描述（要连接的url）

**8.** scrollBehavior滚动行为

使用前端路由，当切换到新路由时，想要页面滚到顶部，或者是保持原先的滚动位置，就像重新加载页面那样。 vue-router 能做到，而且更好，它让你可以自定义路由切换时页面如何滚动。接收to和from两个路由对象

scrollBehavior (to, from, savedPosition) {

// return 期望滚动到哪个的位置

}

**9. @include指令**

使用include指令可以方便的把多个页面中重复显的内容抽取出来，大大的减少代码的重复量，方便我们对重复内容的维护

Vue路由懒加载

const tvProgram = resolve => require(['路由'], resolve);

const game = resolve => require(['路由'], resolve);

const gameitem = resolve => require(['路由'], resolve)

## [require-ensure](http://webpack.github.io/docs/api-in-modules.html#require-ensure)和[require-amd](http://webpack.github.io/docs/api-in-modules.html" \l "require-amd" \t "_blank)的区别：

require-amd

说明: 同AMD规范的require函数，使用时传递一个模块数组和回调函数，模块都被下载下来且都被执行后才执行回调函数

语法: require(dependencies: String[], [callback: function(...)])

参数

dependencies: 模块依赖数组

callback: 回调函数

require-ensure

说明: require.ensure在需要的时候才下载依赖的模块，当参数指定的模块都下载下来了（下载下来的模块还没执行），便执行参数指定的回调函数。require.ensure会创建一个chunk，且可以指定该chunk的名称，如果这个chunk名已经存在了，则将本次依赖的模块合并到已经存在的chunk中，最后这个chunk在webpack构建的时候会单独生成一个文件。

语法: require.ensure(dependencies: String[], callback: function([require]), [chunkName: String])

dependencies: 依赖的模块数组

callback: 回调函数，该函数调用时会传一个require参数

chunkName: 模块名，用于构建时生成文件时命名使用

**注意点**：requi.ensure的模块只会被下载下来，不会被执行，只有在回调函数使用require(模块名)后，这个模块才会被执行

**Vue项目中使用的Mixins**

混入 (mixins) 是一种分发 Vue 组件中可复用功能的非常灵活的方式。混入对象可以包含任意组件选项。当组件使用混入对象时，所有混入对象的选项将被混入该组件本身的选项

**使用：**mixins就是定义一部分公共的方法或者计算属性,然后混入到各个组件中使用,方便管理与统一修改

**1、在assets文件夹下创建一个js文件**

// 创建一个需要混入的对象

export const mixinTest1 = {

created() {

this.hello();

},

methods: {

hello() {

console.log('mixinTest1');

}

}

};

**2、在组件中使用刚刚创建的混入**

import {mixinTest1} from './../assets/js/mixin';

export default {

mixins:[mixinTest1],

name: 'hello',

data () {

return {

msg: 'Welcome to Your Vue.js App'

}

}

}

这样就可以直接调用到混入对象中的hello方法

3、如果组件中定义的方法与混入对象中的方法/属性一样,组件中的优先级大于混入对象中的(方法会调用多次)

4、混入对象中可以定义抽象方法,使用混入的组件必须重写该方法

methods: {

handlePlaylist() {

throw new Error('component must implement handlePlaylist method')

}

}