##### Es6

###### var、let、const之间的区别

var声明变量可以重复声明，而let不可以重复声明  
 var是不受限于块级的，而let是受限于块级  
 var可以在声明的上面访问变量，而let有暂存死区，在声明的上面访问变 量会报错  
 const声明之后必须赋值，否则会报错  
 const定义不可变的量，改变了就会报错  
 const和let一样不会与window相映射、支持块级作用域、在声明的上面访问变量会报错

###### Set、Map的区别？

应用场景Set用于数据重组，Map用于数据储存

Set：　  
 （1）成员不能重复  
 （2）只有键值没有键名，类似数组  
 （3）可以遍历，方法有add, delete,has  
 Map:  
 （1）本质上是健值对的集合，类似集合  
 （2）可以遍历，可以跟各种数据格式转换

###### **Promise 中reject 和 catch 处理上有什么区别**

reject 是用来抛出异常，catch 是用来处理异常  
 reject 是 Promise 的方法，而 catch 是 Promise 实例的方法  
 reject后的东西，一定会进入then中的第二个回调，如果then中没有写第二个回调，则进入catch  
 网络异常（比如断网），会直接进入catch而不会进入then的第二个回调

###### **如何使用Set去重**

Let arr=[1,2,3,2,4,1]

Let newArr = [...new Set(arr)]

###### 理解 async/await以及对Generator的优势

async await 是用来解决异步的，async函数是Generator函数的语法糖  
 使用关键字async来表示，在函数内部使用 await 来表示异步  
 async函数返回一个 Promise 对象，可以使用then方法添加回调函数  
 当函数执行的时候，一旦遇到await就会先返回，等到异步操作完成，再接 着执行函数体内后面的语句

##### vue-router

###### vue-router路由的两种模式

**hash模式：**是VueRouter的默认模式，工作原理是利用hashchange事件监听hash的变化，即#xxx，然后通过。可以在window对象上监听这个事件。

改变hash，浏览器本身不会有任何请求服务器动作的，但是页面状态和url已经关联起来了。

**History模式**：

History模式的书写更加简洁，去掉#。充分利用HTML5 History Interface中新增的pushState()和replaceState()方法，这俩个方法应用于浏览器的历史记录栈（window.history指向History对象，表示当前窗口的浏览历史），使得不刷新页面改变url地址。在已有的back()、forward()、go()（接受一个整数作为参数，若参数超过实际存在的网址范围，该范围无效果；若不指定参数，默认参数为0，相当于刷新页面）的基础上，提供了对历史记录修改的功能。

前进和后退操作时都不会请求服务器，而是通过popstate事件监听，刷新页面才会向服务器发起请求。

**Hash模式的缺点：**

1. hash本来用于做页面定位，若用来做路由，原锚点公共失效。

2. Hash模式的传参基于url，若有复杂的数据会有体积的限制。History模式除了可以利用url传参，还可以将数据存放一个特定的对象中。

3. 路径书写不简约。

Hash模式和History模式的相同点和不同点

相同点：

1. 都是VueRouter的模式配置选项。

2. 都可以用作SPA（单页面应用）的实现，实现前端路由，不向服务器发起请求，动态渲染页面。

不同点：

1. Hash模式需要#xxx（哈希值），History模式书写更简约。

2. Hash模式是通过window对象监听hashchange事件，根据hash值的变化来动态渲染页面的；History模式是通过window对象监听popstate事件，当浏览器前进和后退时，获取当前history对象状态即history.state来动态渲染组件。

3. Hash模式创建history对象是依赖HashHistory构造函数；History模式创建history对象是依赖HTML5History构造函数。

Hash模式利用HashHistory.push()和HashHistory.replace()可以将hash变化的URL都被记录在浏览器历史访问栈，实现页面内容和URL一一对应，从而可以使用浏览器的前进和后退功能。

History模式利用HTML5History.pushState()和HTML5History.replaceState()修改页面的url地址，修改history对象的状态，而不刷新页面。

4. Hash模式通过携带在url中传递参数；History模式既可以通过携带在url中传递参数，也可以将数据存放在一个特定的对象中。

5. Hash模式只有hash值设置为不同于上次时，才会被添加进历史记录栈；History模式可以记录相同的url。

6. Hash模式只能修改#之后的部分；History模式的pushState设置的新的url可以是于当前url同源的任意url

###### 2、vue-router有哪几种导航钩子，导航钩子的作用是什么

详见： https://blog.csdn.net/weixin\_43680581/article/details/104372295

全局导航钩子：router.beforeEach(to,from,next)，router.beforeResolve(to,from,next), router.afterEach(to,from)

组件内的钩子：beforeRouteEnter (to, from, next)

beforeRouteUpdate (to, from, next)

beforeRouteLeave (to, from, next)

单独路由独享组件： beforeEnter: (to, from, next)

作用：vue-router提供的导航钩子主要用来拦截导航，让它完成跳转或取消。

###### 3、$route和$router的区别

$route是“路由信息对象”，包括path，params，hash，query，fullPath，matched，name 等路由信息参数。

$router是“路由实例”对象包括了路由的跳转方法，钩子函数等

this.$router.push() 描述:跳转到不同的url,但这个方法会向history栈添加一个记录,点击 后退会返回到上一个页面。

this.$router.replace() 描述:同样是跳转到指定的url,但是这个方法不会向history里面添 加新的记录,点击返回,会跳转到上上一个页面。上一个记录是不存在的。

##### Vue

###### 1、vue生命周期

指的是从开始创建、初始化数据、编译模板、挂载Dom→渲染、更新→渲染、卸载等一系列过程，我们称这是 Vue 的生命周期

Vue 的生命周期总共分为8个阶段：创建前/后，载入前/后，更新前/后，销毁前/后。

1、beforeCreate（创建前）

表示实例完全被创建出来之前，vue 实例的挂载元素$el和数据对象 data 都为 undefined，还未初始化。

2、created（创建后）

数据对象 data 已存在，可以调用 methods 中的方法，操作 data 中的数据，但 dom 未生成，$el 未存在 。

3、beforeMount（挂载前）

vue 实例的 $el 和 data 都已初始化，挂载之前为虚拟的 dom节点，模板已经在内存中编辑完成了，但是尚未把模板渲染到页面中。data.message 未替换。

4、mounted（挂载后）

vue 实例挂载完成，data.message 成功渲染。内存中的模板，已经真实的挂载到了页面中，用户已经可以看到渲染好的页面了。实例创建期间的最后一个生命周期函数，当执行完 mounted 就表示，实例已经被完全创建好了，DOM 渲染在 mounted 中就已经完成了。

5、beforeUpdate（更新前）

当 data 变化时，会触发beforeUpdate方法 。data 数据尚未和最新的数据保持同步。

6、updated（更新后）

当 data 变化时，会触发 updated 方法。页面和 data 数据已经保持同步了。

7、beforeDestory（销毁前）

组件销毁之前调用 ，在这一步，实例仍然完全可用。

8、destoryed（销毁后）

组件销毁之后调用，对 data 的改变不会再触发周期函数，vue 实例已解除事件监听和 dom绑定，但 dom 结构依然存在。