1. ES6新增的特性

let和const。

箭头函数：

模板字符串

解构赋值

Promise对象

剩余模式

新的数据类型Symbol和Map、Set，

新的方法，如Object.assign、Array.from、Array.find等。

1. 计算属性和监听属性的区别

computed:多对一，支持缓存，不支持异步

watch: 一对多，不支持缓存，支持异步

1. 什么是promise？哪三种状态，实例的哪三个方法分别做了什么？构造函数的哪两个方法及作用

ES6新增的构造函数，可以用来封装异步操作并可以获取其成功/失败的结果值

pedding初始状态 resolve 成功状态 reject 失败状态

then 成功时执行

catch 失败时执行

finally 最后一定会执行

可以解决回调地狱

1. 浅拷贝和深拷贝的区别，如何实现深拷贝

浅拷贝只拷贝一层，深层次的对象只拷贝地址

1.Object.assign()

2.递归

3.lodash库方法 \_cloneDeep

1. v-show和v-if指令的共同点和不同点?

相同点：

都可以根据条件动态地显示或隐藏元素。

在满足条件时，元素会被渲染到DOM中，否则会从DOM中移除。

不同点：

v-show 是通过控制样式来显示隐藏的 display

v-if 是通过控制dom元素来显示隐藏的

v-if 有更高的切换开销

因此，如果需要非常频繁地切换，则使用 v-show 较好

v-if 也是惰性的,如果在初始渲染时条件为假，则什么也不做——直到条件第一次变为真时，才会开始渲染条件块

v-show 有更高的初始渲染开销

如果在运行时条件很少改变，则使用 v-if 较好

1. 组件中传递数据有几种方式？请分别描述下并写出关键语法或关键字

父子传值：props $emit $refs

非父子：bus

1. 字符串的哪些方法

length：返回字符串的长度。

1. indexOf：返回指定字符串在原字符串中第一次出现的位置，如果没有找到则返回-1。
2. slice：提取字符串的一部分，并返回一个新的字符串。
3. substring：和slice类似，提取字符串的一部分，并返回一个新的字符串。接受的参数是起始位置和结束位置。
4. substr：和slice类似，提取字符串的一部分，并返回一个新的字符串。接受的参数是起始位置和要提取的字符数。
5. 使用element UI里面如何覆盖css样式，（v－deep的原理）

使用v-deep指令：v-deep是Vue提供的一种特殊指令，可以用于深度作用选择器，在覆盖Element UI的样式时非常有用。它通过在选择器前添加>>>或/deep/来指示样式规则适用于Element UI组件内部的所有元素

v-deep的原理是使用了深度组合选择器（deep combinator），它可以穿透组件的边界，影响组件内部的所有元素。这是一种强大的样式覆盖方法，但要谨慎使用，以避免样式的溢出和冲突。

1. DOM操作，怎样添加、移除、移动、复制、创建和查找节点？

添加节点: 可以使用createElement()方法创建一个新的节点，然后使用appendChild()方法将其添加到指定的父节点上

移除节点: 可以使用removeChild()方法从父节点中移除指定的子节点。

移动节点: 可以使用appendChild()方法将已有的节点移动到新的父节点上。

复制节点: 可以使用cloneNode()方法复制一个节点，然后使用appendChild()方法将其添加到目标父节点上

创建节点: 可以使用createElement()方法创建一个新的节点，然后使用appendChild()方法将其添加到父节点上

查找节点: 可以使用querySelector()和querySelectorAll()方法根据选择器查找符合条件的节点。

1. 双向绑定的原理

Vue 数据双向绑定原理是通过 `数据劫持` + `发布者-订阅者模式` 的方式来实现的，首先是通过 `ES5` 提供的 `Object.defineProperty()` 方法来劫持（监听）各属性的 \*\*getter、setter\*\*，并在当监听的属性发生变动时通知订阅者，是否需要更新，若更新就会执行对应的更新函数。

1. 同步和异步的区别 （那些是异步

同步 / 异步

// ? 同步： 一件事情一件事情做

// ? 异步： 一件事情做一半，去做另外一件事情

1. 异步函数：例如setTimeout、setInterval函数。
2. 异步请求：例如Ajax请求、fetch请求等网络请求。
3. 异步读写：例如读取文件、写入数据库等IO操作。
4. 事件监听：例如DOM事件、键盘事件等事件处理。
5. echarts折线图如何让用户看起来数据更新的比较平滑

使用动画效果：在echarts中，可以为图表设置动画效果，使得数据更新的过程更加平缓。可以通过设置animation参数来控制动画效果的时长和缓动函数。可以适当延长动画的时长，使得数据更新的过程更加缓慢，从而给用户产生平滑的感觉。

使用平滑曲线：除了使用直线来连接数据点外，还可以尝试使用平滑曲线（如贝塞尔曲线）来连接数据点，使得整体的趋势更加平缓。在echarts中，可以通过设置smooth参数为true来启用平滑曲线。

调整数据更新的频率：如果数据更新的频率过高，可能会给用户造成不稳定或抖动的感觉。可以适当控制数据的更新频率，使得变化更加平缓。可以考虑使用缓存技术，将数据按照一定的时间间隔更新到图表上，或者使用滚动窗口的方式展示数据。

使用渐变效果：除了直接更新数据点，还可以考虑使用渐变效果，在每次更新数据后，将旧的数据点逐渐淡出，新的数据点逐渐淡入，这样可以让整个图表的变化看起来更加平滑。

1. 如何让CSS只在当前组件中起作用

在style标签中加入scoped

1. vuex 是什么?怎么使用?哪种功能场景使用它?

Vuex是一个专为Vue.js应用程序开发的状态管理模式库。它集中管理应用程序的状态，使得状态的变化可以被追踪，共享和管理，从而简化了组件间的通信和状态管理。

在Vue.js应用中使用Vuex需要先安装并导入Vuex模块，并在应用的根组件中创建一个Vuex的store，将其作为Vue实例的store选项。然后，在需要使用store的组件中，可以通过this.$store来访问store中的状态和方法。

Vuex适用于以下场景：

1. 多个组件共享同一个状态，需要对状态进行集中的管理和更新。
2. 组件之间需要进行状态的同步和共享，方便组件之间的通信。
3. 管理大型项目的状态，使得状态变化更加可控和易于维护。
4. 处理异步操作和复杂的状态流，方便管理和追踪异步操作的状态。
5. flex 有那些属性

flex-direction：指定弹性容器的主轴方向，可以是row（默认，水平方向），row-reverse（反向水平方向），column（垂直方向），column-reverse（反向垂直方向）。

justify-content：定义了弹性容器中子项在主轴上的对齐方式。常用取值有flex-start（默认，左对齐），flex-end（右对齐），center（居中对齐），space-between（两端对齐，项目之间间隔相等），space-around（每个子项两侧的间隔相等）。

align-items：定义了弹性容器中子项在交叉轴上的对齐方式。常用取值有flex-start（上对齐），flex-end（下对齐），center（居中对齐），baseline（基线对齐），stretch（默认，子项在交叉轴上被拉伸以填满剩余空间）。

flex-wrap：设置弹性容器的子项是否换行。常用取值有nowrap（默认，不换行），wrap（换行），wrap-reverse（反向换行）。

align-content：定义了多根轴线的对齐方式（当弹性容器有多行时）。常用取值有flex-start（与交叉起始对齐），flex-end（与交叉结束对齐），center（居中对齐），space-between（两端对齐，轴线之间间隔相等），space-around（每个轴线两侧的间隔相等），stretch（默认，轴线被拉伸以填满剩余空间）。

flex-grow：定义了弹性子项的放大比例。默认值为0，表示不放大。如果存在多个子项，且为正值，则按照比例进行分配剩余空间。

flex-shrink：定义了弹性子项的缩小比例。默认值为1，表示收缩的比例与前者相同。

flex-basis：指定弹性子项在主轴方向上的初始大小。

flex：是flex-grow、flex-shrink和flex-basis的简写形式。

1. vue的router路由模式有几种

Vue-Router有两个路由模式，分别是哈希hash模式和历史history模式，然后默认的是哈希hash模式。

1. 数组的循环map() 和 forEach()的区别,返回值

forEach()自行遍历 无返回值

map() 有返回值

1. get和post的区别

1、从语义上来看，GET为检索和获取，而POST的意思为创建或更新。 GET多用于从服务器请求获取数据，POST多用于向服务器发送数据。 2、一些在w3cschool里的标准答案： GET在浏览器回退时是无害的，而POST会再次提交请求

1. keep-alive的作用是什么?

在组件切换过程中将状态保留在内存中，防止重复渲染DOM，减少加载时间及性能消耗，提高用户体验性。