**js篇**

1、

浏览器模型，渲染原理，JS解析过程，JS运行机制

**一web storage 和cookie有什么区别？**

共同点：都是保存在浏览器端,且同源的

区别：

cookie作为http协议的一部份，cookie数据在浏览器和服务器间来回传递

存储大小限制也不同

cookie数据不能超过4K，webStorage可以达到5M或更大

数据的有效期不同

  sessionStorage：仅在当前的浏览器窗口关闭有效；

      localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；

      cookie：只在设置的cookie过期时间之前一直有效，即使窗口和浏览器关闭

作用域不同

      sessionStorage：不在不同的浏览器窗口中共享，即使是同一个页面；

      localStorage：在所有同源窗口都是共享的；

        cookie：也是在所有同源窗口中共享的

**二javascipt实现继承有哪些？**

1. 原型链继承。2、构造继承。3、实例继承。4、拷贝继承。

5、组合继承 6、寄生组合继承

**四举列内存泄露和解决方法**

1．意外的全局变量一个未声明的变量的引用在全局对象中创建了一个新变量

这些全局变量是无法进行垃圾回收的，对其赋值为 null 或者重新分配。

2．被忘记的 Timers 或者 callbacks

3．闭包

4．DOM 引用

**六．vue组件的通信方案有哪些**

### 1.父子组件之间的消息传递 props down, events up

### 父组件向子组件传递数据是通过prop传递的，

### 子组件传递数据给父组件是通过$emit触发事件来做到的

### 2.兄弟组件通信 创建一个Vue的实例bus, bus.$emit()

### 3.vuex处理组件之间的数据交互

### 七.DOM事件的绑定有哪几种方式

1.在DOM元素中直接绑定；

2.在JavaScript代码中绑定；

3.绑定事件监听函数。

### 八VUE keep-alive的应用场景是什么

### keep-alive 是Vue的内置组件，能在组件切换过程中将状态保留在内存中，防止重复渲染DOM。结合vue-router中使用，可以缓存某个view的整个内容。

**面试题**

**一this指向：**

this误解：1.this引用function本身

2.this引用的是function的词法作用域

this机制的四种规则：

1.默认绑定全局变量(window)

2.隐式绑定(函数被作为某个对象的方法调用时)

3.显式绑定（call( )）

4.new新对象绑定

**二new操作符干了以下三步:**

1.先创建了一个新的空对象

2.然后让这个空对象的\_\_proto\_\_指向函数的原型prototype

3.将对象作为函数的this传进去，如果return 出来东西是对象的话就直接返回 return 的内容，没有的话就返回创建的这个对象

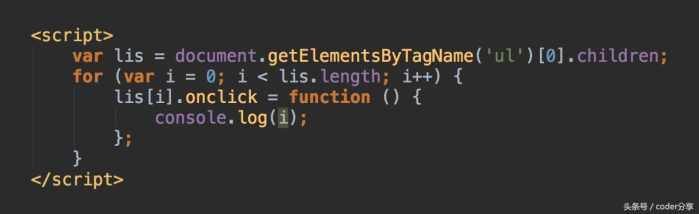
**三闭包：**

闭包是指有权访问另一个 函数作用域中的变量的函数。创建闭包的常见方式，就是在一个函数内部创建另一个函数,保护变量不被改变（内部函数的作用域链仍然保持着对父函数活动对象的引用）

由于闭包会携带包含它的函数的作用域，因此会比其他函数占用更多的内存。过 度使用闭包可能会导致内存占用过多，我们建议读者只在绝对必要时再考虑使用闭 包。虽然像 V8 等优化后的 JavaScript 引擎会尝试回收被闭包占用的内存，但请大家 还是要慎重使用闭包。

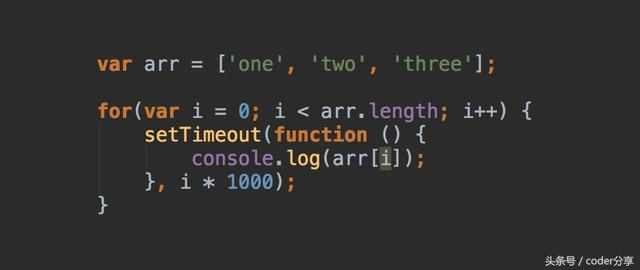
Eg:

ul中有若干个li，每次点击li，输出li的索引值：

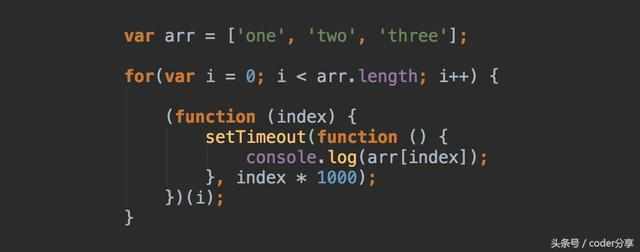


在我们点击li，触发li的click事件之前，for循环已经执行结束了，此时内存中的i只有一个值，那就是i=lis.length，所以每次点击li后返回的都是lis.length的值





在setTimeout的函数执行时，for循环已经结束，此时i=4，而arr[4]=undefined，所以三次都是返回undefined



**四原型链**

原型对象：就是在函数创建的时候,创建了一个它的实例对象并赋值给它的prototype->object

原型链作为实现继承的主要方法，就是通过原型对象实现的，其基本思想是利用原 型让一个引用类型继承另一个引用类型的属性和方法(new) 这样，子类型就能够访问父类型的所有属性和方法

**五.作用域和作用域链**

作用域：全局变量和局部变量

定义在函数里面变量称为局部变量，定义在函数外面的称为全局变量，不过有一个注意需要注意的是，函数内部声明变量的时候，一定要使用var命令。如果不用的话，你实际上声明了一个全局变量

作用域链：

根据在内部函数可以访问外部函数变量的这种机制，用链式查找决定哪些数据能被内部函数访问。-》执行环境

每个函数运行时都会产生一个执行环境，**每一个执行环境关联了一个**内部属性，该内部属性包含了函数被创建的作用域中对象的集合，这个集合被称为函数的作用域链，它决定了哪些数据能被函数访问。

**六.什么是跨域请求？如何实现跨域请求？**

**<http://www.ruanyifeng.com/blog/2016/04/cors.html>**

当前URL与目标URL的“协议名或主机名或端口”三者中有一项不同，目标URL请求就称为“跨域请求”。

浏览器允许跨域的情形： <img><script><link><iframe>

浏览器禁止跨域的情形： XHR请求

XHR跨域被禁止的解决方案：

1. 图片ping+script标签
2. JSONP
3. CORS跨域资源共享
4. Window.name+iframe
5. Window.postMessage
6. WebSocket
7. 代理
8. Document.domain

七请谈谈http协议

在了解http协议之前要先了解5层网络模型，这五层分别是应用层、传输层、网络层、数据链路层、物理层。http相关的主要有应用层和传输层，应用层用到的协议是http协议，传输层的是TCP/IP协议。Http主要经过了0.9 1.0 .1.1 2.0这三个版本，现在常有用的是1.1版本，http是超文本传输协议，是请求响应模型，主要通过数据包的，要传输数据包，就要借助传输层中的TCP/IP协议，其中最经典的是三次握手

**九．请描述一下javascript 事件冒泡机制**

当事件被触发时，由当前被触发的元素沿着DOM树，一层层往外寻找要触发的事件，找到绑定有对应事件的元素时，就会触发该事件，这样由内往外寻找触发事件元素，一直找到根节点

**十．网页渲染过程**

1. 构建DOM树
2. 构建CSSOM树
3. 执行JavaScript

4. 构建渲染树

5. 布局

6. 绘制

**十二.重排和重绘**

浏览器在渲染完首屏页面后，如果对DOM进行操作会引起浏览器引擎对DOM渲染树的重新布局和重新绘制，称之为“重排”和“重绘”(一个元素外观的改变所触发的浏览器行为，例如改变visibility、outline、背景色)，由于重排和重绘有前后的依赖关系，重绘发生时不一定会引起重排，但是如果发生重排时必定会引起重绘。所以要避免重排，减少重绘。

减少重排的方法：

1. 将多次改变样式属性的操作合并成一次操作，例如：用class+js操作

2. 将需要多次重排的元素，position属性设为absolute或fixed，这样此元素就脱离了文档流

3. 由于display属性为none的元素不在渲染树中，对隐藏的元素操作不会引发其他元素的重排。如果要对一个元素进行复杂的操作时，可以先隐藏它，操作完成后再显示。这样只在隐藏和显示时触发2次重排。

**十四. 浏览器中输入一个网址到网页内容完全展示发生的事情？**

1. 浏览器会开启一个单独的线程来处理这个指令，首先要判断用户输入的是否是一个合法或是合理的URL地址，如果是就进行下一步。
2. 浏览器引擎对URL进行解析，如果存在缓存(cache-control)且未过期，则会从本地缓存中提取文件；如果缓存不存在或是缓存已过期就发起远程请求。
3. 通过DNS解析域名获取该网址对应的IP地址，连同浏览器的cookie、userAgent等信息向此IP发出GET请求。
4. 接下来就是经典的三次握手，http协议会话，浏览器客户端向web服务器发送报文，进行通讯和数据传输。
5. 进入网站的后台服务。
6. 服务器根据相应的URL执行相应的后端应用逻辑，期间会使用到服务器端缓存或是数据库。
7. 服务器端处理请求并返回响应的报文，如果浏览器访问过该页面，缓存有对应资源，与服务器最后的修改对比，一致就返回304，否则返回200和对应的内容。
8. 浏览器收到返回信息并开始下载该HTML文件(无缓存，200返回码)或从本地缓存提取文件(有缓存，304返回码)。
9. 浏览器的渲染引擎在拿到HTML文件之后，开始解析构建DOM树，并根据HTML中的标记请求下载指定的MIME类型文件（CSS、JavaScript脚本等），同时使用&设置缓存等内容。
10. 渲染引擎根据CSS样式规则将DOM树扩充为渲染树，然后进行重排、重绘。
11. 如果有JS文件将会执行，进行DOM操作，缓存杜村，事件绑定等操作。最终在页面上显示出来。

**2018-11-19**

**简要说明一下vue的生命周期**

四个阶段，8个函数 create(创建) mount(挂载) update(更新) destroy(销毁)

**计算属性(computed)跟侦听器(watch)有什么不同**

计算属性：就是通过其它值计算另一个值，计算属性具有缓存，就是当依赖的值发生改变

才会进行重新计算，一般用于简单的逻辑处理

侦听器：侦听一个特定的值，当这个值发生变化时才执行特定的值

**用户刷新后，vuex state数据会丢失？**

数据保存在本地，localstorage.setItem(key,val)

**v-model的原理?**

v-model是实现双向数据绑定，但它单单是一个语法糖，它是依靠:value 跟@input来实现

<input v-model=’something’/>

||

<input :value=’ something’ @input=” something=$event.target.value”/>

**Axios默认携带cookie不？**

axios的访问请求默认是不带cookie的，因此想要带cookie的请求，

需要每次请求时设置这个参数：withCredentials: true；

也可以全局设置 : axios.defaults.withCredentials=true

**登录拦截？**

https://www.jianshu.com/p/269ac710fa81

**Axios的拦截器？**

https://www.jianshu.com/p/269ac710fa81

请求拦截器、响应拦截器

**1.let与var的区别？**

let 为 ES6 新添加申明变量的命令，它类似于 var，但是有以下不同：

var 声明的变量，其作用域为该语句所在的函数内，且存在变量提升现象

let 声明的变量，其作用域为该语句所在的代码块内，不存在变量提升

let 不允许重复声明.

**2.为什么 var 可以重复声明？**

当我们执行代码时，我们可以简单的理解为新变量分配一块儿内存，命名为a，

并赋值为2，但在运行的时候编译器与引擎还会进行两项额外的操作：

判断变量是否已经声明：

首先编译器对代码进行分析拆解，从左至右遇见var a，则编译器会询问作用域是否已经存在叫 a 的变量了，如果不存在，则招呼作用域声明一个新的变量a，若已经存在，则忽略var 继续向下编译，这时a = 2被编译成可执行的代码供引擎使用。

引擎遇见a=2时同样会询问在当前的作用域下是否有变量a，若存在，则将a赋值为2（由于第一步编译器忽略了重复声明的var，且作用域中已经有a，所以重复声明会发生值得覆盖而并不会报错）。若不存在，则顺着作用域链向上查找，若最终找到了变量a则将其赋值2，若没有找到，则招呼作用域声明一个变量a并赋值为2。

**怎么判断两个json对象的内容相等？**

JSON.stringify（深拷贝与浅拷贝）

**重排和重绘**

部分渲染树（或者整个渲染树）需要重新分析并且节点尺寸需要重新计算。这被称为重排。注意这里至少会有一次重排-初始化页面布局。

由于节点的几何属性发生改变或者由于样式发生改变，例如改变元素背景色时，屏幕上的部分内容需要更新。这样的更新被称为重绘。

**什么情况会触发重排和重绘？**

添加、删除、更新 DOM 节点

display: none 隐藏一个 DOM 节点-触发重排和重绘

通过 visibility: hidden 隐藏一个 DOM 节点-只触发重绘，因为没有几何变化

移动或者给页面中的 DOM 节点添加动画

添加一个样式表，调整样式属性

用户行为，例如调整窗口大小，改变字号，或者滚动。

**浏览器缓存**

浏览器缓存分为强缓存和协商缓存。当客户端请求某个资源时，获取缓存的流程如下：

先根据这个资源的一些 http header 判断它是否命中强缓存，如果命中，则直接从本地获取缓存资源，不会发请求到服务器；

当强缓存没有命中时，客户端会发送请求到服务器，服务器通过另一些request header验证这个资源是否命中协商缓存，称为http再验证，如果命中，服务器将请求返回，但不返回资源，而是告诉客户端直接从缓存中获取，客户端收到返回后就会从缓存中获取资源；

强缓存和协商缓存共同之处在于，如果命中缓存，服务器都不会返回资源；

区别是，强缓存不对发送请求到服务器，但协商缓存会。

当协商缓存也没命中时，服务器就会将资源发送回客户端。

当 ctrl+f5 强制刷新网页时，直接从服务器加载，跳过强缓存和协商缓存；

当 f5 刷新网页时，跳过强缓存，但是会检查协商缓存；

防抖和节流

所谓防抖，就是指触发事件后，就是把触发非常频繁的事件合并成一次去执行。即在指定时间内只执行一次回调函数

所谓节流，是指频繁触发事件时，只会在指定的时间段内执行事件回调，即触发事件间隔大于等于指定的时间才会执行回调函数。

### 什么是 mvvm？

MVVM 是 Model-View-ViewModel 的缩写。mvvm 是一种设计思想。Model 层代表数据模型，也可以在 Model 中定义数据修改和操作的业务逻辑；View 代表 UI 组件，它负责将数据模型转化成 UI 展现出来，ViewModel 是一个同步 View 和 Model 的对象。

在 MVVM 架构下，View 和 Model 之间并没有直接的联系，而是通过 ViewModel 进行交互，Model 和 ViewModel 之间的交互是双向的， 因此 View 数据的变化会同步到 Model 中，而 Model 数据的变化也会立即反应到 View 上。

ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，而 View 和 Model 之间的同步工作完全是自动的，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作 DOM, 不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。

### mvvm 和 mvc 区别？

mvc 和 mvvm 其实区别并不大。都是一种设计思想。主要就是 mvc 中 Controller 演变成 mvvm 中的 viewModel。mvvm 主要解决了 mvc 中大量的 DOM 操作使页面渲染性能降低，加载速度变慢，影响用户体验。和当 Model 频繁发生变化，开发者需要主动更新到 View 。

### vue生命周期的理解？

* 创建前/后： 在 beforeCreate 阶段，vue 实例的挂载元素 el 还没有。
* 载入前/后：在 beforeMount 阶段，vue 实例的$el 和 data 都初始化了，但还是挂载之前为虚拟的 dom 节点，data.message 还未替换。在 mounted 阶段，vue 实例挂载完成，data.message 成功渲染。
* 更新前/后：当 data 变化时，会触发 beforeUpdate 和 updated 方法。
* 销毁前/后：在执行 destroy 方法后，对 data 的改变不会再触发周期函数，说明此时 vue 实例已经解除了事件监听以及和 dom 的绑定，但是 dom 结构依然存在

### vue-router 有哪几种导航钩子?

### vuex 是什么？怎么使用？哪种功能场景使用它？

**几个常见的loader**

file-loader：把文件输出到一个文件夹中，在代码中通过相对 URL 去引用输出的文件

url-loader：和 file-loader 类似，但是能在文件很小的情况下以 base64 的方式把文件内容注入到代码中去

source-map-loader：加载额外的 Source Map 文件，以方便断点调试

image-loader：加载并且压缩图片文件

babel-loader：把 ES6 转换成 ES5

css-loader：加载 CSS，支持模块化、压缩、文件导入等特性

style-loader：把 CSS 代码注入到 JavaScript 中，通过 DOM 操作去加载 CSS。

eslint-loader：通过 ESLint 检查 JavaScript 代码

**axios的特点有哪些？**

一、Axios 是一个基于 promise 的 HTTP 库，支持promise所有的API

二、它可以拦截请求和响应

三、它可以转换请求数据和响应数据，并对响应回来的内容自动转换成 JSON类型的数据

四、安全性更高，客户端支持防御 XSRF

**2.axios有哪些常用方法？**

一、axios.get(url[, config]) //get请求用于列表和信息查询

二、axios.delete(url[, config]) //删除

三、axios.post(url[, data[, config]]) //post请求用于信息的添加

四、axios.put(url[, data[, config]]) //更新操作

**什么是webpack**

webpack是一个打包模块化javascript的工具，在webpack里一切文件皆模块，通过loader转换文件，通过plugin注入钩子，最后输出由多个模块组合成的文件，webpack专注构建模块化项目。

WebPack可以看做是模块打包机：它做的事情是，分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其打包为合适的格式以供浏览器使用。

官网的图片形象的展示了webpack的定义

**.如何阻止Vue中的绑定事件不发生冒泡**

可以使用“事件修饰符”来处理事件冒泡，如：v-on:click.stop阻止事件冒泡

或v-on:submit.prevent阻止默认事件。

**为什么组件中的data属性的值必须是一个函数？**

因为在一个组件被多次引用的情况下，如果data的值是一个Object的话，那么由于Object是一个引用类型，所以即使是该组件被多次引用，而其实操作的是同一个对象，最终导致了引用该组件的所有位置都同步的显示了。

**7.非父子层级的组件如何实现通信？**

简单的应用场景下，可以使用一个空的Vue实例作为中央事件总线。

在复杂的情况下，可以考虑使用Vue 官方提供的状态管理模式——Vuex来进行管理。