有道 youdao

2021年前端大厂面试真题

—— 源码解析篇

悄悄变强大

網易 NETEASE www·163·com

网易云课堂

谨以此书献给所有辛勤努力的同学，他们的积极分享是我们创作灵感的源泉。祝早日拿到心仪的OFFER。

—— 永生

“天才是1%的灵感，99%的汗水，但那1%的灵感是最重要的，甚至比那99%的汗水都要重要。

—— 李春阳

本书在写作风格上推陈新出，对于用例的讲解，不仅文字描述，更以事例佐证。我们咬文嚼字，字斟句酌，所有的这些付出，只为能让读者对书中的技术点放心，文字描述舒心，感受编程的魅力之处。

—— 陈赟帆

有道 youdao

目录

有道 youdao

Javascript

第 一 章

01 函数

02 异步编程

03 设计模式

01 函数

考点 1 对于闭包的了解以及使用闭包的好处你知道哪些? 难度系数: ★★★★★

【网易有道事业部/ 京东电子文娱事业部】

网易专家解析:

闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，创建的函数可以访问到当前函数的局部变量。

闭包有两个常用的用途:

1、使我们在函数外部能够访问到函数内部的变量。通过使用闭包，我们可以通过在外部调用闭包函数，从而在外部访问到函数内部的变量，可以使用这种方法来创建私有变量。

2、使已经运行结束的函数上下文中的变量对象继续留在内存中，因为闭包函数保留了这个变量对象的引用，所以这个变量对象不会被回收。其实闭包的本质就是作用域链的一个特殊的应用，只要了解了作用域链的创建过程，就能够理解闭包的实现原理。

考点 2 谈谈你对于 This 对象的了解? 难度系数: ★★★★★

【百度信息技术工程事业部/ 阿里社区电商事业部】

网易专家解析:

This 对象包括以下四种模式:

1、第一种是函数调用模式，当一个函数不是一个对象的属性时，直接作为函数来调用时，this 指向全局对象。

2、第二种是方法调用模式，如果一个函数作为一个对象的方法来调用时，this 指向这个对象。

3、第三种是构造器调用模式，如果一个函数用 new 调用时，函数执行前会新创建一个对象，this 指向这个新创建的对象。

4.第四种是 apply 、 call 和 bind 调用模式，这三个方法都可以显示的指定调用函数的 this 指向。其中 apply 方法接收两个参数：一个是 this 绑定的对象，一个是参数数组。call 方法接收的参数，第一个是 this 绑定的对象，后面的其余参数是传入函数执行的参数。也就是说，在使用 call() 方法时，传递给函数的参数必须逐个列举出来。bind 方法通过传入一个对象，返回一个 this 绑定了传入对象的新函数。这个函数的 this 指向除了使用 new 时会被改变，其他情况下都不会改变。

考点 3 请简述你对于 JavaScript 作用域链的了解? **难度系数:** ★★★★☆

【腾讯微信事业部 / 百度移动生态事业部】

网易专家解析:

作用域链的作用是保证对执行环境有权访问的所有变量和函数的有序访问，通过作用域链，我们可以访问到外层环境的变量和函数。作用域链的本质上是一个指向变量对象的指针列表。变量对象是一个包含了执行环境中所有变量和函数的对象。作用域链的前端始终都是当前执行上下文的变量对象。全局执行上下文的变量对象（也就是全局对象）始终是作用域链的最后一个对象。当我们查找一个变量时，如果当前执行环境中没有找到，我们可以沿着作用域链向后查找。

考点 4 举例说明 stopPropagation 能阻止事件冒泡，还能阻止事件捕获传递。

难度系数: ★★★★★

【网易有道事业部/ 京东电子文娱事业部】

网易专家解析:

如果在捕获阶段，调用了 stopPropagation，则按事件流的顺序，后面的捕获和冒泡事件都不会执行，同样，只在冒泡阶段调用，只能阻止后面的冒泡事件执行。

```
<div id="root">
  <button id="btn">点我</button>
```

```
</div>
<script>
  var root = document.getElementById("root");
  var btn = document.getElementById("btn");
  root.addEventListener("click", function (e) {
    e.stopPropagation()
    console.log("root-捕获")
  }, true);
  root.addEventListener("click", function (e) {
    console.log("root-冒泡")
  });
  btn.addEventListener("click", function (e) {
    console.log("btn-捕获")
  }, true);
  btn.addEventListener("click", function (e) {
    console.log("btn-冒泡")
  });
</script>
```

02 异步编程

考点 5 谈谈你对于浏览器同源政策的了解有哪些?

难度系数: ★★★★☆

【百度 web 前端事业部/ 京东 3C 数码家电事业部】

网易专家解析:

对浏览器的同源政策的理解是，一个域下的 JavaScript 脚本在未经允许的情况下，不能够访问另一个域的内容。这里的同源的指的是两个域的协议、域名、端口号必须相同，否则则不属于同一个域。同源政策主要限制了三个方面:

1、当前域下的 JavaScript 脚本不能够访问其他域下的 Cookie、localStorage 和 indexDB。

2、当前域下的 JavaScript 脚本不能够操作访问操作其他域下的 DOM。

3、是当前域下 Ajax 无法发送跨域请求。

考点 6 解决 JavaScript 的加载、解析和执行会阻塞页面渲染过程问题的方法有哪些?

难度系数: ★★★★★

【阿里社区电商事业部/ 百度信息技术工程部】

网易专家解析:

JavaScript 的加载、解析和执行会阻塞页面的渲染过程，因此我们希望 JavaScript 脚本能够尽可能的延迟加载，提高页面的渲染速度。解决此类问题有以下四种方法:

1、我们一般采用的是将 JavaScript 脚本放在文档的底部，来使 JavaScript 脚本尽可能的在最后来加载执行。

2、给 JavaScript 脚本添加 defer 属性，这个属性会让脚本的加载与文档的解析同步解析，然后在文档解析完成后再执行这个脚本文件，这样的话就能使页面的渲染不被阻塞。多个设置了 defer 属性的脚本按规范来说最后是顺序执行的，但是在一些浏览器中可能不是这样。

3、给 JavaScript 脚本添加 Async 属性，这个属性会使脚本异步加载，不会阻塞页面的解析过程，但是当脚本加载完成后立即执行 JavaScript 脚本，这个时候如果文档没有解析完成的话同样会阻塞。多个 Async 属性的脚本的执行顺序是不可预测的，一般不会按照代码的顺序依次执行。

4、动态创建 DOM 标签的方式，我们可以对文档的加载事件进行监听，当文档加载完成后再动态的创建 script 标签来引入 JavaScript 脚本。

考点 7 对于 Use Strict 的理解以及使用它的好处和坏处有哪些?

难度系数：★★★★★

【百度信息技术工程事业部】

网易专家解析：

在代码中出现表达式 Use Strict 意味着代码按照严格模式解析，这种模式使得 JavaScript 在更严格的条件下运行。

好处：

1、消除 JavaScript 语法的一些不合理、不严谨之处，减少一些怪异行为；

2、消除代码运行的一些不安全之处，保证代码运行的安全；

3、提高编译器效率，增加运行速度；

4、为未来新版本的 JavaScript 做好铺垫。

坏处：

1、同样的代码，在"严格模式"中，可能会有不一样的运行结果；

2、一些在"正常模式"下可以运行的语句，在"严格模式"下将不能运行。

03 设计模式

考点 8 **谈谈对于 JavaScript 事件循环的了解?** **难度系数:** ★★★★☆

【腾讯技术工程事业部/ 小米互动娱乐事业部】

网易专家解析:

首先 JavaScript 是单线程运行的，在代码执行的时候，通过将不同函数的执行上下文压入执行栈中来保证代码的有序执行。在执行同步代码的时候，如果遇到了异步事件，JavaScript 引擎并不会一直等待其返回结果，而是会将这个事件挂起，继续执行执行栈中的其他任务。当同步事件执行完毕后，再将异步事件对应的回加入到与当前执行栈中不同的另一个任务队列中等待执行。任务队列可以分为宏任务对列和微任务对列，当当前执行栈中的事件执行完毕后，JavaScript 引擎首先会判断微任务对列中是否有任务可以执行，如果有就将微任务队首的事件压入栈中执行。当微任务对列中的任务都执行完成后再去判断宏任务对列中的任务。

考点 9 **通过代码呈现防抖和节流功能函数。** **难度系数:** ★★★★★

【腾讯微信事业部 / 百度移动生态事业部】

网易专家解析:

代码如下:

```
// 节流
function throttle(fn,delay){
    let timer
    return function() {
      if(timer){
          return false
```


```
        }
        timer = setTimeout(() => {
          fn()
          clearTimeout(timer);
          timer = null;
        }, delay)
    }
}
// 防抖
function debounce(fn,delay){
    let timer = null
    return function() {
        if(timer){
            clearTimeout(timer)
        }
        timer = setTimeout(fn,delay)
    }
}
```

考点 10 ES6 模块与 CommonJavaScript 模块、AMD、CMD 的差异有哪些?

难度系数: ★★★★☆

【百度 AI 技术平台事业部】

网易专家解析:

ES6 模块与 CommonJavaScript 模块、AMD、CMD 的差异有以下两点:

1、CommonJavaScript 模块输出的是一个值的拷贝，ES6 模块输出的是值的引用。

CommonJavaScript 模块输出的是值的拷贝，也就是说，一旦输出一个值，模块内部的变化就影响不到这个值。ES6 模块的运行机制与 CommonJavaScript 不一样。JavaScript 引擎对脚本静态分析的时候，遇到模块加载命令 import，就会生成一个只读引用。等到脚本真正执行时，再根据这个只读引用，到被加载的那个模块里面去取值。

2、CommonJavaScript 模块是运行时加载，ES6 模块是编译时输出接口。CommonJavaScript 模块就是对象，即在输入时是先加载整个模块，生成一个对象，然后再从这个对象上面读取方法，这种加载称为“运行时加载”。而 ES6 模块不是对象，它的对外接口只是一种静态定义，在代码静态解析阶段就会生成。

有道 youdao

Vue

第 二 章

01 深入Vue源码设计
02 TypeScript实战
03 Vue3.x的设计理念
04 Vue3.x API设计
05 Vue4.x API设计

01 深入 Vue 源码设计

考点 11 **谈谈对于 Vue 中异步更新队列的理解。** **难度系数：★★★★★**

【阿里产业电商事业部】

网易专家解析：

Vue 在更新 DOM 时是异步执行的。只要侦听到数据变化，Vue 将开启一个队列，并缓冲在同一事件循环中发生的所有数据变更。如果同一个 watcher 被多次触发，只会被推入到队列中一次。这种在缓冲时去除重复数据对于避免不必要的计算和 DOM 操作是非常重要的。然后在下一个的事件循环 “tick” 中，Vue 刷新队列并执行实际 (已去重的) 工作。Vue 在内部对异步队列尝试使用原生的 Promise.then、MutationObserver 和 setImmediate，如果执行环境不支持，则会采用 setTimeout(fn, 0) 代替。

02 TypeScript 实战

考点 12 **请介绍一下你对 TypeScript 中泛型的了解?** **难度系数:★★★★☆**

【百度 web 前端事业部 / 阿里社区电商事业部】

网易专家解析:

TypeScript 泛型是一个提供创建可重用组件方法的工具。它能够创建可以处理多种数据类型而不是单一数据类型的组件。泛型在不影响性能或生产率的情况下提供类型安全性。泛型允许我们创建泛型类、泛型函数、泛型方法和泛型接口。在泛型中，类型参数写在开(<)和闭(>)括号之间，这使得它是强类型集合。泛型使用一种特殊类型的类型变量<T>，它表示类型。泛型集合只包含类似类型的对象。

03 Vue3.x 的设计理念

考点 13 在 Vue 开发中，SPA 单页面应用十分广泛，请简述它的优缺点分别是什么？

难度系数：★★★★☆

【字节跳动互动娱乐事业部 / 快手电商事业部】

网易专家解析：

SPA 单页面的优点：

1、用户体验好、快，内容的改变不需要重新加载整个页面，避免了不必要的跳转和重复渲染，对服务器压力小；

2、前后端职责分离，架构清晰，前端进行交互逻辑，后端负责数据处理。

SPA 单页面的缺点：

1、初次加载耗时多，为实现单页 web 应用功能及展示效果，需要在加载页面时将 JS、CSS 统一加载，部分页面按需加载；

2、前进后退路由管理：由于单页应用在一个页面中显示所有的内容，所以不能使用浏览器的前进后退功能，所有的页面切换需要自己建立堆栈管理。

考点 14 Vue 实际开发中应该在哪个生命周期内调用异步请求?

难度系数: ★★★★★

【百度 web 前端事业部 / 京东 3C 数码家电事业部】

网易专家解析:

可以在钩子函数 created、beforeMount、mounted 中进行调用，因为在这三个钩子函数中，data 已经创建，可以将服务端端返回的数据进行赋值。比较推荐在 created 钩子函数中调用异步请求，因为在 created 钩子函数中调用异步请求有以下优点:

1、能更快获取到服务端数据，减少页面 loading 时间;

2、ssr 不支持 beforeMount 、mounted 钩子函数，所以放在 created 中有助于一致性。

考点 15 在实际开发中，Vue 中的 key 有什么作用?

难度系数: ★★★★☆

【百度信息技术工程事业部 / 阿里社区电商事业部】

网易专家解析:

key 是为 Vue 中 vnode 的唯一标记。Vue 的 Diff 过程可以概括为：oldCh 和 newCh 各有两个头尾的变量 oldStartIndex、oldEndIndex 和 newStartIndex、newEndIndex，它们会新节点和旧节点会进行两两对比，如果设置了 key，就会用 key 再进行比较，在比较的过程中，遍历会往中间靠，一旦 StartIdx > EndIdx 表明 oldCh 和 newCh 至少有一个已经遍历完了，就会结束比较，通过这个 key，我们的 Diff 操作可以更准确、更快速，在 sameNode 函数 a.key === b.key 对比中可以避免就地复用的情况。

04 Vue3.x API 设计

考点 16 虚拟 DOM 是前端性能优化的一大利器，请简单描述其优缺点有哪些?

难度系数：★★★★☆

【百度移动互联网事业部 / 阿里云智能事业部】

网易专家解析：

虚拟 DOM 的优点：

1、保证性能下限： 框架的虚拟 DOM 需要适配任何上层 API 可能产生的操作，它的一些 DOM 操作的实现必须是普适的，所以它的性能并不是最优的；但是比起粗暴的 DOM 操作性能要好很多，因此框架的虚拟 DOM 至少可以保证在你不需要手动优化的情况下，依然可以提供还不错的性能，即保证性能的下限；

2、无需手动操作 DOM： 我们不再需要手动去操作 DOM，只需要写好 View-Model 的代码逻辑，框架会根据虚拟 DOM 和 数据双向绑定，帮我们以可预期的方式更新视图，极大提高我们的开发效率；

3、跨平台： 虚拟 DOM 本质上是 JavaScript 对象，而 DOM 与平台强相关，相比之下虚拟 DOM 可以进行更方便地跨平台操作，例如服务器渲染、weex 开发等等。

虚拟 dom 的缺点：

1、无法进行极致优化： 虽然虚拟 DOM + 合理的优化，足以应对绝大部分应用的性能需求，但在一些性能要求极高的应用中虚拟 DOM 无法进行针对性的极致优化。

考点 17 Vue 中双向数据绑定是如何实现的?

难度系数：★★★★☆

【腾讯云与智慧产业事业部】

网易专家解析：

Vue 双向数据绑定是通过数据劫持、结合、发布订阅模式的方式来实现的，也就是说数据和视图同步，数据发生变化，视图跟着变化，视图变化，数据也随之发生改变。关于 Vue 双向数据绑定，其核心是 Object.defineProperty()方法。

05 Vue4.x API 设计

考点 18 Axios 是什么？怎样使用它？怎么解决跨域的问题？ **难度系数：★★★★★**

【腾讯微信事业部】

网易专家解析：

Axios 的是一种异步请求，用法和 Ajax 类似，安装 npm install Axios --save 即可使用，请求中包括 get，post，put，patch，delete 等五种请求方式，解决跨域可以在请求头中添加 Access-Control-Allow-Origin，也可以在 index.js 文件中更改 proxyTable 配置等解决跨域问题。

有道 youdao

React

第 三 章

01 探索React正殿
02 React生态应用
03 React原理
04 React状态管理
05 React高级实践和性能优化

01 探索 React 正殿

考点 19 如何理解 React 中的生命周期函数?

难度系数: ★★★★★

【腾讯网络媒体事业部 / 阿里产业电商事业部】

网易专家解析:

React 的生命周期函数可以概括为下列几个阶段:

1、初始化阶段:

getDefaultProps:获取实例的默认属性

getInitialState:获取每个实例的初始化状态

componentWillMount:组件即将被装载、渲染到页面

Render:组件在这里生成虚拟的 Dom 节点

componentDidMount:组件真正在被装载之后

2、运行中状态:

componentWillReceiveProps:组件将要接收到属性的时候调用

shouldComponentUpdate:组件接受到新属性或者新状态的时候(可以返回 false,接收数据后不更新,阻止 Render 调用,后面的函数不会被继续执行了)

componentWillUpdate:组件即将更新不能修改属性和状态

Render:组件重新描绘

componentDidUpdate:组件已经更新

3、销毁阶段:

componentWillUnmount:组件即将销毁

02 React 生态应用

考点 20 如果进行三次 setState 会发生什么，循环执行 setState 组件会一直重新渲染吗?

难度系数：★★★★☆

【网易雷火事业部】

网易专家解析：

关于 setState 的使用描述如下：

1、setState 这个方法是用来告诉 React 组件数据有更新，有可能需要重新渲染；

2、它是异步的，React 通常会集齐一批需要更新的组件，然后一次性更新来保证渲染的性能，所以进行三次只有一次的效果，并不会导致多次渲染；

3、此外再使用 setState 改变状态之后，立刻使用 this.State 去拿最新的状态往往是拿不到的，如果需要最新的 State 做业务的话，可以在 componentDidUpdate 或者 setState 的回调函数里获取。（官方推荐第一种）。

03 React 原理

考点 21 如何理解 React 中的 Diff 原理?

难度系数: ★★★★★

【美团互联网云事业部】

网易专家解析:

React 中的 Diff 原理简单描述如下:

1、把树形结构按照层级分解，只比较同级元素，给列表结构的每个单元添加唯一的 Key 属性，方便比较;

2、React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）;

3、合并操作，调用 component 的 setState 方法的时候， React 将其标记为 dirty.到每一个事件循环结束，React 检查所有标记 dirty 的 component 重新绘制;

4、选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 Diff 的性能。

考点 22 React 实现的移动应用中如果出现卡顿有哪些可以考虑的优化方案?

难度系数: ★★★★★

【阿里社区电商事业部】

网易专家解析:

优化方案如下:

1、增加 shouldComponentUpdate 钩子对新旧 Props 进行比较如果值相同则阻止更新避免不必要的渲染;

2、对于列表或其他结构相同的节点为其中的每一项增加唯一 Key 属性以方便 React 的 Diff 算法中对该节点的复用，减少节点的创建和删除操作;

3、Render 函数中减少类似 onClick={() => {doSomething()}}的写法每次调用 Render 函数时均会创建一个新的函数即使内容没有发生任何变化也会导致节点没必要的重渲染，建议将函数保存在组件的成员对象中这样只会创建一次；

4、组件的 Props 如果需要经过一系列运算后才能拿到最终结果可以考虑使用 Reselect 库对结果进行缓存，如果 Props 值未发生变化则结果直接从缓存中取得，避免高昂的运算代价；

5、webpack-bundle-analyzer 分析当前页面的依赖包是否存在不合理性如果存在找到优化点并进行优化。

04 React 状态管理

考点 23 Redux 有哪些优点?

难度系数: ★★★★☆

【腾讯网络媒体事业部】

网易专家解析:

Redux 的优点简单描述如下:

1、结果的可预测性 - 由于总是存在一个真实来源，即 Store ，因此不存在如何将当前状态与动作和应用的其他部分同步的问题;

2、可维护性 - 代码变得更容易维护，具有可预测的结果和严格的结构;

3、服务器端渲染 - 你只需将服务器上创建的 Store 传到客户端即可。这对初始渲染非常有用，并且可以优化应用性能，从而提供更好的用户体验;

4、开发人员工具 - 从操作到状态更改，开发人员可以实时跟踪应用中发生的所有事情;

5、易于测试 - Redux 的代码主要是小巧、纯粹和独立的功能。这使代码可测试且独立;

6、组织 - Redux 准确地说明了代码的组织方式，这使得代码在团队使用时更加一致和简单。

05 React 高级实践和性能优化

考点 24 如何理解 React-Fiber，它解决了什么问题？ **难度系数：★★★★☆**

【快手商业化事业部】

网易专家解析：

React-Fiber 简单描述如下：

1、React V15 在渲染时，会递归比对 VirtualDom 树，找出需要变动的节点，然后同步更新它们， 一气呵成。这个过程期间， React 会占据浏览器资源，这会导致用户触发的事件得不到响应，并且会导致掉帧，导致用户感觉到卡顿；

2、为了给用户制造一种应用很快的“假象”，不能让一个任务长期霸占着资源。可以将浏览器的渲染、布局、绘制、资源加载(例如 HTML 解析)、事件响应、脚本执行视作操作系统的“进程”，需要通过某些调度策略合理地分配 CPU 资源，从而提高浏览器的用户响应速率， 同时兼顾任务执行效率。

React 通过 Fiber 架构，让这个执行过程变成可被中断。“适时”地让出 CPU 执行权，除了可以让浏览器及时地响应用户的交互，还有其他好处：

1、分批延时对 Dom 进行操作，避免一次性操作大量 Dom 节点，可以得到更好的用户体验；

2、给浏览器一点喘息的机会，它会对代码进行编译优化（JIT）及进行热代码优化，或者对 reflow 进行修正。

考点 25 在 React 中如何避免不必要的 Render?

难度系数: ★★★★★

【百度移动互联网事业部 / 阿里云智能事业部】

网易专家解析:

React 基于虚拟 Dom 和高效 Diff 算法的完美配合，实现了对 Dom 最小粒度的更新。大多数情况下，React 对 Dom 的渲染效率足以解决日常业务。但在个别复杂业务场景下，性能问题依然会困扰我们，可以从以下几点优化:

1、shouldComponentUpdate 和 PureComponent:

在 React 类组件中，可以利用 shouldComponentUpdate 或者 PureComponent 来减少因父组件更新而触发子组件的 Render，从而达到目的。shouldComponentUpdate 来决定是否组件是否重新渲染，如果不希望组件重新渲染，返回 false 即可。

2、利用高阶组件:

在函数组件中，并没有 shouldComponentUpdate 这个生命周期，可以利用高阶组件，封装一个类似 PureComponet 的功能。

3、使用 React.memo:

React.memo 是 React 16.6 新的一个 API，用来缓存组件的渲染，避免不必要的更新，其实也是一个高阶组件，与 PureComponent 十分类似，但不同的是， React.memo 只能用于函数组件。

考点 26 React 中的虚拟 Dom 是怎么实现的?

难度系数: ★★★★★

【小米互动娱乐事业部腾讯云 / 智慧产业事业部】

网易专家解析:

首先说说为什么要使用 Virturl Dom，因为操作真实 Dom 的耗费的性能代价太高所以 React 内部使用 JS 实现了一套 Dom 结构。

在每次操作真实 Dom 之前使用 Diff 算法对虚拟 Dom 进行比较递归找出有变化的 Dom 节点然后对其进行更新操作。为了实现虚拟 Dom 我们需要把每一种节点类型抽象成对象。每一种节点类型有自己的属性即 Prop。每次进行 Diff 的时候 React 会先比较该节点类型，如果节点类型不一样那么 React 会直接删除该节点然后直接创建新的节点插入到其中，如果节点类型一样那么会比较 Prop 是否有更新，如果 Prop 不一样，那么 React 会判定该节点有更新那么重渲染该节点然后在对其子节点进行比较一层一层往下直到没有子节点。

考点 27 axios 是什么？怎样使用它？怎么解决跨域的问题？

难度系数：★★★★★

【腾讯微信事业部】

网易专家解析：

axios 的是一种异步请求，用法和 ajax 类似，安装 npm install axios --save 即可使用，请求中包括 get，post，put，patch，delete 等五种请求方式，解决跨域可以在请求头中添加 Access-Control-Allow-Origin，也可以在 index.js 文件中更改 proxyTable 配置等解决跨域问题。

有道 youdao

Node.js

第 四 章

01 Web服务及Koa

02 企业级Node.js框架

01 Web 服务及 Koa

考点 28 谈谈你对于内存泄漏的理解?

难度系数: ★★★★☆

【网易雷火事业部】

网易专家解析:

内存泄漏(Memory Leak)指由于疏忽或错误造成程序未能释放已经不再使用的内存的情况。

如果内存泄漏的位置比较关键，那么随着处理的进行可能持有越来越多的无用内存，这些无用的内存变多会引起服务器响应速度变慢。

严重的情况下导致内存达到某个极限(可能是进程的上限，如 V8 的上限；也可能是系统可提供的内存上限)会使得应用程序崩溃。

考点 29 对于定位内存泄漏的排查方法有哪些?

难度系数: ★★★★☆

【百度信息技术工程事业部 / 阿里社区电商事业部】

网易专家解析:

想要定位内存泄漏，通常会有两种情况:

1、对于只要正常使用就可以重现的内存泄漏，这是很简单的情况只要在测试环境模拟就可以排查了。

2、对于偶然的内存泄漏，一般会与特殊的输入有关系。想稳定重现这种输入是很耗时的过程。

如果不能通过代码的日志定位到这个特殊的输入，那么推荐去生产环境打印内存快照了。

需要注意的是，打印内存快照是很耗 CPU 的操作，可能会对线上业务造成影响。快照工具推荐使用 heapdump 用来保存内存快照，使用 devtool 来查看内存快照。使用 heapdump 保存内存快照时，只会有 Node 环境中的对象，不会受到干扰(如果使用 Node-inspector 的话，快照中会有前端的变量干扰)。

考点 30 对于创建子进程方法的理解，以及它们之间的区别有哪些?

难度系数：★★★★★

【腾讯微信事业部 / 百度移动生态事业部】

网易专家解析：

创建子进程的方法有以下 6 种：

1、spawn()： 启动一个子进程来执行命令；

2、exec()：启动一个子进程来执行命令，与 spawn()不同的是其接口不同，它有一个回调函数获知子进程的状况；

3、execFlie()：启动一个子进程来执行可执行文件；

4、fork()：与 spawn()类似，不同电在于它创建 Node 子进程需要执行 js 文件；

5、spawn()与 exec()、execFile()不同的是，后两者创建时可以指定 timeout 属性设置超时时间，一旦创建的进程超过设定的时间就会被杀死；

6、exec()与 execFile()不同的是，exec()适合执行已有命令，execFile()适合执行文件。

02 企业级 Node.js 框架

考点 31 请谈谈你对于 Buffer 内存分配机制的理解?

难度系数: ★★★★☆

【阿里云智能事业部】

网易专家解析:

为了高效的使用申请来的内存，Node 采用了 slab 分配机制。slab 是一种动态的内存管理机制。Node 以 8kb 为界限来来区分 Buffer 为大对象还是小对象，如果是小于 8kb 就是小 Buffer，大于 8kb 就是大 Buffer。

例如第一次分配一个 1024 字节的 Buffer，Buffer.alloc(1024)，那么这次分配就会用到一个 slab，接着如果继续 Buffer.alloc(1024)，那么上一次用的 slab 的空间还没有用完，因为总共是 8kb，1024+1024 = 2048 个字节，没有 8kb，所以就继续用这个 slab 给 Buffer 分配空间。如果超过 8kb，那么直接用 C++底层地宫的 SlowBuffer 来给 Buffer 对象提供空间。

考点 32 Node 的优缺点是什么?

难度系数: ★★★★☆

【阿里云智能事业部】

网易专家解析:

优点如下:

1、Node 是基于事件驱动和无阻塞的，非常适合处理并发请求，因此构建在 Node 的代理服务器相比其他技术实现的服务器要好一点;

2、与 Node 代理服务器交互的客户端代码由 JavaScript 语言编写，客户端与服务端都采用一种语言编写;

缺点如下:

1、Node .js 是一个相对新的开源项目，不太稳定，变化速度快;

2、不适合 CPU 密集型应用，如果有长时间运行的计算（比如大循环），将会导致 CPU 时间片不能释放，使得后续 I/O 无法发起。

有道 youdao

工程化

第 五 章

01 持续集成与部署

02 Git操作

01 持续集成与部署

考点 33 如何理解 WebPack 与 Grunt、Gulp 的不同之处?

难度系数:★★★★☆

【字节跳动互动娱乐事业部】

网易专家解析:

1、Grunt、Gulp 是基于任务运行的工具:

它们会自动执行指定的任务，就像流水线，把资源放上去然后通过不同插件进行加工，它们包含活跃的社区，丰富的插件，能方便的打造各种工作流。

2、WebPack 是基于模块化打包的工具:

自动化处理模块，WebPack 把一切当成模块，当 WebPack 处理应用程序时，它会递归地构建一个依赖关系图(dependency graph)，其中包含应用程序需要的每个模块，然后将所有这些模块打包成一个或多个 Bundle;

因此这是完全不同的两类工具，而现在主流的方式是用 npm script 代替 Grunt、Gulp，npm script 同样可以打造任务流。

02 Git 操作

考点 34 提交时发生冲突，你是如何解决的?

难度系数：★★★★☆

【美团互联网云事业部 / 阿里产业电商事业部】

网易专家解析：

1、发生冲突，在 IDE 里面一般都是对比本地文件和远程分支的文件，然后把远程分支上文件的内容手工修改到本地文件，然后再提交冲突的文件使其保证与远程分支的文件一致，这样才会消除冲突，然后再提交自己修改的部分。特别要注意下，修改本地冲突文件使其与远程仓库的文件保持一致后，需要提交后才能消除冲突，否则无法继续提交。必要时可与同事交流，消除冲突；

2、通过 git stash 命令，把工作区的修改提交到栈区，目的是保存工作区的修改；

3、通过 git pull 命令，拉取远程分支上的代码并合并到本地分支，目的是消除冲突；

4、通过 git stash pop 命令，把保存在栈区的修改部分合并到最新的工作空间中。

考点 35 WebPack 的构建流程是什么?

难度系数：★★★★☆

【百度智能生活事业部】

网易专家解析：

WebPack 的运行流程是一个串行的过程，从启动到结束会依次执行以下流程：

1、初始化参数：从配置文件和 Shell 语句中读取与合并参数，得出最终的参数；

2、开始编译：用上一步得到的参数初始化 Compiler 对象，加载所有配置的插件，执行对象的 run 方法开始执行编译；

3、确定入口：根据配置中的 entry 找出所有的入口文件；4、编译模块：从入口文件出发，调用所有配置的 Loader 对模块进行翻译，再找出该模块依赖的模块，再递归本步骤直到所有入口依赖的文件都经过了本步骤的处理；

5、完成模块编译：在经过第 4 步使用 Loader 翻译完所有模块后，得到了每个模块被翻译后的最终内容以及它们之间的依赖关系；

6、输出资源：根据入口和模块之间的依赖关系，组装成一个个包含多个模块的 Chunk，再把每个 Chunk 转换成一个单独的文件加入到输出列表，这步是可以修改输出内容的最后机会；

7、输出完成：在确定好输出内容后，根据配置确定输出的路径和文件名，把文件内容写入到文件系统。

有道 youdao

计算机网络

第 六 章

01 网络协议

02 网络请求实战

03 网络安全与攻防

01 网络协议

考点 36 请说出 ISO 七层模型以及对他们的理解? **难度系数:★★★★☆**

【字节跳动互动娱乐事业部】

网易专家解析:

ISO 七层模型是国际标准化组织制定的一个用于计算机或通信系统间互联的标准体系。以下是对这七层模型的理解:

1、应用层:网络服务与最终用户的一个接口,常见的协议有:HTTP FTP SMTP SNMP DNS;

2、表示层:数据的表示、安全、压缩。,确保一个系统的应用层所发送的信息可以被另一个系统的应用层读取。

3、会话层:建立、管理、终止会话,对应主机进程,指本地主机与远程主机正在进行的会话;

4、传输层:定义传输数据的协议端口号,以及流控和差错校验,协议有 TCP UDP;

5、网络层:进行逻辑地址寻址,实现不同网络之间的路径选择,协议有 ICMP IGMP IP 等;

6、数据链路层:在物理层提供比特流服务的基础上,建立相邻结点之间的数据链路;

7、物理层:建立、维护、断开物理连接。

考点 37 你知道哪些可以预防 SQL 注入问题的方法?

难度系数: ★★★★☆

【百度智能生活事业部】

网易专家解析:

1、使用#{}而不是 ${}，在 MyBatis 中，使用#{}而不是${}，可以很大程度防止 sql 注入。

因为#{}是一个参数占位符，对于字符串类型，会自动加上""，其他类型不加。由于 Mybatis 采用预编译，其后的参数不会再进行 SQL 编译，所以一定程度上防止 SQL 注入。${}是一个简单的字符串替换，字符串是什么，就会解析成什么，存在 SQL 注入风险；

2、不要暴露一些不必要的日志或者安全信息，比如避免直接响应一些 sql 异常信息。如果 SQL 发生异常了，不要把这些信息暴露响应给用户，可以自定义异常进行响应；

3、不相信任何外部输入参数，过滤参数中含有的一些数据库关键词关键词。可以加个参数校验过滤的方法，过滤 union，or 等数据库关键词；

4、适当的权限控制。在你查询信息时，先校验下当前用户是否有这个权限。比如说，实现代码的时候，可以让用户多传一个企业 Id 什么的，或者获取当前用户的 session 信息等，在查询前，先校验一下当前用户是否是这个企业下的等等，是的话才有这个查询员工的权限。

02 网络请求实战

考点 38 **请谈谈你对快速重传的理解以及它的流程是什么？** **难度系数：★★★★☆**

【腾讯网络媒体事业部 / 阿里产业电商事业部】

网易专家解析：

快速重传可以用来解决超时重发的时间等待问题，它不以时间驱动，而是以数据驱动。它是基于接收端的反馈信息来引发重传的。快速重传流程如下：

发送方发送了 1，2，3，4，5，6 份数据；

第一份 Seq=1 先送到了，于是就 Ack 回 2；

第二份 Seq=2 也送到了，于是 ACK 回 3；

第三份 Seq=3 由于网络等某些原因，没送到；

第四份 Seq=4 送到了，但是由于 Seq=3 没收到。因此 ACK 还是回 3；

后面的 Seq=5，6 的也送到了，ACK 还是回复 3，因为 Seq=3 没有收到；

发送方连着收到三个重复冗余 ACK=3 的确认，于是知道哪个报文段在传输过程中丢失了；发送方在定时器过期之前，重传该报文段；

最后，接收方收到了 Seq=3，此时因为 Seq=4，5，6 都收到了，于是它回 ACK=7。

考点 39 **对于如下程序的运行结果是什么？**

```
setTimeout(() => {
 console.log(1);
 Promise.resolve().then(() => {
   console.log(2);
 });
```


```
}, 0);
new Promise((resolve) => {
  console.log(3);
  resolve();
}).then(() => {
  console.log(4);
});
Promise.resolve()
  .then(() => {
    console.log(5);
  })
  .then(() => {
    console.log(6);
    setTimeout(() => {
      console.log(7);
    }, 1000);
  })
  .then(() => {
    console.log(8);
  });
```

难度系数: ★★★★★

【腾讯网络媒体事业部 / 阿里产业电商事业部】

网易专家解析：

该程序为浏览器事件环问题，运行结果为： 3 4 5 6 8 1 2 7

原因如下：

1、遇到 setTimeout 交给 setTimeout 的线程执行，定时器是 0 秒，线程立即把回调放到宏任务队列中；

2、遇到 new Promise，注意 new Promise 的回调是立即执行的，先打印出 3，把 then 放到微任务队列中；

3、遇到 Promise.resolve，把 5 放到微任务队列中；

4、主代码执行完毕，接着清空微任务；

5、微任务队列中，第一个放进去的是 4，打印出 4；

6、第二个微任务是 5，打印出 5，在执行 5 时，又遇到微任务 6，加入到尾部队列中，接着执行 6 的微任务，打印出 6；

7、微任务 6 执行时，遇到 setTimeout，交给 setTimeout 线程处理，setTimeout 线程会在 1 秒后把回调放到宏任务队列中；

8、微任务 6 执行完，遇到微任务 8，同理也会一块清空 8，打印出 8；

9、此时微任务清空完毕，拿出一个宏任务执行，打印 1；

10、宏任务 1 执行时，遇到微任务 2，加入微任务队列中；

11、接着清空所有微任务，打印 2；

12、setTimeout 线程 1 秒后，把 7 加入宏任务队列中，最后会打印出 7。

03 网络安全与功防

考点 40 你知道哪些可以预防 SQL 注入问题的方法? **难度系数:** ★★★★★

【百度智能生活事业部】

网易专家解析:

1、使用#{}而不是 ${}，在 MyBatis 中，使用#{}而不是${}，可以很大程度防止 sql 注入。因为#{}是一个参数占位符，对于字符串类型，会自动加上""，其他类型不加。由于 Mybatis 采用预编译，其后的参数不会再进行 SQL 编译，所以一定程度上防止 SQL 注入。${}是一个简单的字符串替换，字符串是什么，就会解析成什么，存在 SQL 注入风险;

2、不要暴露一些不必要的日志或者安全信息，比如避免直接响应一些 sql 异常信息。如果 SQL 发生异常了，不要把这些信息暴露响应给用户，可以自定义异常进行响应;

3、不相信任何外部输入参数，过滤参数中含有的一些数据库关键词关键词。可以加个参数校验过滤的方法，过滤 union，or 等数据库关键词;

4、适当的权限控制。在你查询信息时，先校验下当前用户是否有这个权限。比如说，实现代码的时候，可以让用户多传一个企业 Id 什么的，或者获取当前用户的 session 信息等，在查询前，先校验一下当前用户是否是这个企业下的等等，是的话才有这个查询员工的权限。

网易云课堂

悄悄变强大