2024美团前端面试题

1. 实习经历介绍，你项目里面有封装button，你是怎么设计的，有什么功能
2. css 水平和垂直居中的实现
3. event loop
4. 闭包(为什么?在内存中怎么存?)
5. 异步的方法有哪些，讲一下 promsie
6. promise.all() 和 promise.race()不同
7. ES module 和 commonJS
8. composition API有哪些
9. vue3父组件和子组件之间传值有哪些方法(为代码考察)10.状态管理 用了

pinia 简单讲一下

10. 输入url到最后渲染的流程12.介绍http缓存(强缓存，协商缓存)
11. 算法题:合并排序数组

答案：

1. 实习经历中的Button封装设计

在封装Button组件时，我会考虑将按钮的所有必要功能和样式变得可配置。这包括按钮文本、颜色、大小、是否禁用、点击处理等。例如，在React中，我可以创建一个Button组件，使用props传递这些参数，并使用条件渲染处理不同的状态，如加载状态或禁用状态。代码如下：

```
function Button({ text, onClick, disabled, size, color }) {
  return (
```

```
    <button
      onClick={onClick}
      disabled={disabled}
      style={{ fontSize: size, backgroundColor: color }}
    >
      {text}
    </button>
  );
}
```

2. CSS水平和垂直居中的实现

Flexbox方法是最简单直观的方式：

```
.container {
  display: flex;
  justify-content: center;
  align-items: center;
}
使用Grid布局：
css
Copy code
.container {
  display: grid;
  place-items: center;
}
```

3. Event Loop

JavaScript是单线程的，Event Loop使得它可以执行异步操作。它工作原理是，主线程运行的代码会放入一个调用栈中，异步事件（如setTimeout, 网络请求等）会被放入任务队列。只有当调用栈为空时，Event Loop会从任务队列中取出任务执行。

4. 闭包

闭包允许函数访问其定义时作用域中的变量，即使外部函数已经执行完毕。在内存中，闭包会保持对这些外部变量的引用，这就是为什么这些变量不会被垃圾回收器回收。闭包是实现模块化和私有变量的重要方式。

5. 异步方法和Promise

异步方法包括回调函数、Promise和async/await。Promise是一种支持异步操作的对象，可以用.then()方法链式调用处理异步结果，用.catch()处理错误。

一个Promise实例:

```
const promise = new Promise((resolve, reject) => {
  if (/* condition */) {
    resolve(value);
  } else {
    reject(error);
  }
});
```

6. Promise.all()和Promise.race()

Promise.all()等待所有给定的promise全部完成，它返回一个promise，该promise成功回调的结果是一个数组，包含所有promise的结果。如果任一promise失败，返回的promise立即拒绝。

Promise.race()返回一个promise，它解决或拒绝与第一个解决或拒绝的输入promise相同。

7. ES Module和CommonJS

ES Module使用import和export，支持静态分析和静态优化，如tree-shaking。

CommonJS使用require()和module.exports，它是运行时加载，因此无法进行一些编译时优化。

8. Vue 3 Composition API

Vue 3的Composition API包括setup()函数，里面可以使用ref, reactive, computed, watch等来创建和管理响应式状态，使得功能按逻辑更好组织。

9. Vue3父子组件通信

父向子传值通过props，子向父通信可以通过$emit触发事件。

示例代码：

```
<!-- Parent.vue -->
<template>
  <Child :data="parentData" @childEvent="handleChildEvent"/>
</template>
```

10. Pinia状态管理

Pinia提供了定义和管理全局状态的方式，它类似于Vuex但更简单。状态通过stores管理，每个store是一个响应式对象。

12. HTTP缓存：强缓存与协商缓存

HTTP缓存是一种强大的功能，用于减少网络延迟，提高网站性能，减少服务器负载。它可以分为两种类型：强缓存和协商缓存。

强缓存

强缓存不会向服务器发送请求，直接从缓存中读取资源。它是通过HTTP响应头中的Cache-Control和Expires来控制的。

Cache-Control: 这是HTTP/1.1中引入的，它的值如max-age=3600表示资源在3600秒内都是新鲜的，不需要重新请求。

Expires: 这是HTTP/1.0的产物，提供一个具体的日期/时间，之后缓存的资源被认为过期（不推荐使用，因为它依赖于本地时间可能与服务器时间不同步）。

使用强缓存时，如果缓存未过期，浏览器将不会联系服务器，用户会立即看到加载的页面，从而大大提高效率。

协商缓存

当强缓存失效后，浏览器会与服务器进行通信，检查文件是否被修改，这称为协商缓存。这是通过Last-Modified/If-Modified-Since和ETag/If-None-Match响应和请求头实现的。

Last-Modified/If-Modified-Since: 服务器响应包含Last-Modified标头，标示资源最后修改时间。浏览器后续请求相同资源时，发送If-Modified-Since头，包含相同的日期。如果服务器上的资源自那日期未改变，服务器响应304 Not Modified，资源从浏览器缓存加载。

ETag/If-None-Match: ETag是资源的特定版本的标识符。浏览器在请求头中使用If-None-Match发送ETag值。如果资源未更改，服务器返回304状态码，资源从缓存中加载。

协商缓存虽然需要与服务器进行通信，但如果资源未更改，通过发送很小的信息（304响应），而不是完整的资源，仍可节省带宽和加载时间。

13. 算法题：合并排序数组

合并两个排序数组是经典的算法问题，通常出现在面试中。这里我们讨论的是将两个已排序的数组合并成一个新的已排序数组的问题。

方法

最直观的方法是使用双指针技术。创建两个指针分别指向两个数组的开始，比较两个指针指向的元素，将较小的元素添加到结果数组中，并移动相应的指针。重复这一过程直到一个数组被完全消耗。最后，如果有剩余的元素，直接将它们追加到结果数组的末尾。

示例代码（Python）

```python
def merge_sorted_arrays(arr1, arr2):
    merged = []
    i, j = 0, 0
    while i < len(arr1) and j < len(arr2):
        if arr1[i] <= arr2[j]:
            merged.append(arr1[i])
            i += 1
        else:
            merged.append(arr2[j])
            j += 1
    # 添加剩余元素
    if i < len(arr1):
        merged.extend(arr1[i:])
    if j < len(arr2):
        merged.extend(arr2[j:])
    return merged
```

这种方法的时间复杂度是O(n + m)，其中n和m分别是两个数组的长度，因为每个元素只被访问一次。空间复杂度是O(n + m)，用于存储合并后的