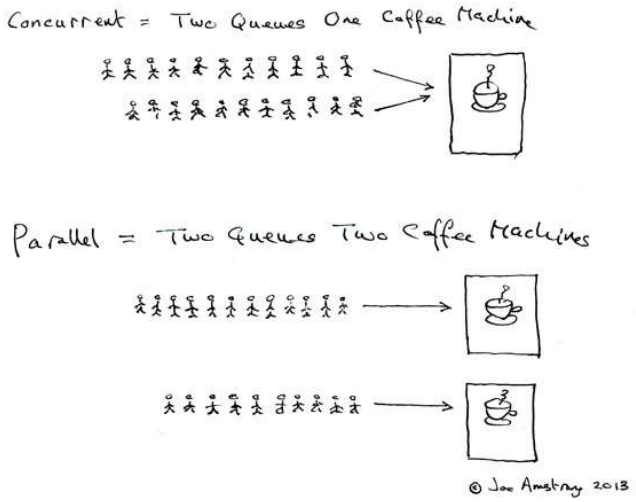
面试内容补充

2021年5月14日 12:54

并发和并行

并发的关键是你有处理多个任务的能力，不一定要同时。
并行的关键是你有同时处理多个任务的能力。
所以我认为它们最关键的点就是：是否是『同时』。



JS的严格模式

它不是一条语句，但是是一个字面量表达式，在 JavaScript 旧版本中会被忽略。
"use strict" 的目的是指定代码在严格条件下执行。
严格模式下你不能使用未声明的变量。

为什么是三次握手而不是2次，4次

这是为了防止无效的连接请求报文到达B服务机。

因为有可能A先发了一个连接请求报文，但是由于网络的1问题，迟迟没有到达B主机，这时候，A主机就超时重传了该报文，然后B主机响应了该请求报文，但是不妙了，第一个报文居然又到了B主机，那么B主机就会把它作为新的连接请求，如果只有两次握手，那么B主机对于该连接请求也会建立连接，但是如果是三次握手，B主机发出确认报文后，A主机不予理睬，这样就不会建立TCP连接了。

所以说只有当三次握手完成，也就是A主机发出确认报文的时候，双方主机才会进入ESTABLISHED状态。

预加载的几种实现方法

- 使用HTML标签

```
<img src="http://pic26.nipic.com/20121213/6168183 0044449030002.jpg" style="display:none"/>
```

- 使用Image对象

```
<script src="./myPreload.js"></script>
```

```
//myPreload.js文件
 var image= new Image()
 image.src="http://pic26.nipic.com/20121213/6168183 004444903000 2.jpg"
```

登录后复制

- 使用XMLHttpRequest对象，虽然存在跨域问题，但会精细控制预加载过程

```
var xmlhttprequest=new XMLHttpRequest();
xmlhttprequest.onreadystatechange=callback;
xmlhttprequest.onprogress=progressCallback;
xmlhttprequest.open("GET","http://image.baidu.com/mouse,jpg",true);
xmlhttprequest.send();
function callback(){
  if(xmlhttprequest.readyState==4&& xmlhttprequest.status==200){
    var responseText=xmlhttprequest.responseText;
  }else{
     console.log("Request was unsuccessful:"+xmlhttprequest.status);
  }
}
function progressCallback(e){
e=e || event;
if(e.lengthComputable){
console.log("Received"+e.loaded+"of"+e.total+"bytes")
}
}

```

- 使用PreloadJS库
  PreloadJS提供了一种预加载内容的一致方式，以便在HTML应用程序中使用。预加载可以使用HTML标签以及XHR来完成。默认情况下，PreloadJS会尝试使用XHR加载内容，因为它提供了对进度和完成事件的更好支持，但是由于跨域问题，使用基于标记的加载可能更好。

```
//使用preload.js
var queue=new createjs.LoadQueue();//默认是xhr对象，如果是new createjs.LoadQueue(false)是指使用HTML标签，可以跨域
queue.on("complete",handleComplete,this);
queue.loadManifest([
{id:"myImage",src:"http://pic26.nipic.com/20121213/6168183  0044449030002.jpg"},
{id: "myImage2", src:"http://pic9.nipic.com/20100814/2839526  1931471581702.jpg"}
]);
function handleComplete(){
  var image=queue.getResuLt("myImage");
  document.body.appendChild(image);
}
```

懒加载实现方式

1.第一种是纯粹的延迟加载，使用setTimeOut或setInterval进行加载延迟.
2.第二种是条件加载，符合某些条件，或触发了某些事件才开始异步下载。
3.第三种是可视区加载，即仅加载用户可以看到的区域，这个主要由监控滚动条来实现，一般会在距用户看到某图片前一定距离遍开始加载，这样能保证用户拉下时正好能看到图片。

```javascript
//javascript
<script type="text/javascript">
window.onload = function(){
    var imgs = Array.from(document.querySelectorAll('img'));
    delay();
    window.addEventListener('scroll', exec(delay, 4000));

    function exec(fun, delaytime){
         var timeout,startTime = new Date();
        return function(){
            clearTimeout(timeout);
            tiemout = setTimeout(fun, delaytime);
        }
    }

    function delay(){
        var scrollTop = window.scrollY;
        var clientHeight = document.documentElement.clientHeight;
        //console.log(clientHeight);
        imgs.forEach((item, index)=>{
            if(scrollTop==0 && item.offsetTop<=clientHeight+scrollTop){
                //console.log(item.getAttribute('data_img'));
                item.setAttribute('src',item.getAttribute('data_img'));
            }else if(item.offsetTop<=clientHeight+scrollTop && item.offsetTop > scrollTop){
                item.setAttribute('src',item.getAttribute('data_img'));
            }
        });
    }
}
</script>
```

object.create()和new的区别

new关键字创建的对象会保留原构造函数的属性，而用Object.create()创建的对象不会

instanceof和isPrototypeOf

A.isPrototypeOf(B)： A对象是否存在于B对象的原型链之中

A instanceof B： B.prototype是否存在与A的原型链之中

Async和await

async 用于申明一个 function 是异步的，而 await 用于等待一个异步方法执行完成。

async 函数返回一个 Promise 对象，可以使用 then 方法添加回调函数。

async 函数中可能会有 await 表达式，async 函数执行时，如果遇到 await 就会先暂停执行，等到触发的异步操作完成后，恢复 async 函数的执行并返回解析值。

await 关键字仅在 async function 中有效。如果在 async function 函数体外使用 await，你只会得到一个语法错误。

virtual DOM

用javascript对象结构表示DOM树结构

在页面进行更新的时候，借助VD，DOM元素的改变可以在内存中进行比较，再结合框架的事务机制将多次比较的结果合并后一次性更新到页面，从而有效地减少页面渲染的次数，提高渲染效率。

DOM树的形成

- 如果压入到栈中的是StartTag Token，HTML 解析器会为该 Token 创建一个 DOM 节点，然后将该节点加入到 DOM 树中，它的父节点就是栈中相邻的那个元素生成的节点。
- 如果分词器解析出来是文本 Token，那么会生成一个文本节点，然后将该节点加入到 DOM 树中，文本 Token 是不需要压入到栈中，它的父节点就是当前栈顶 Token 所对应的 DOM 节点。
- 如果分词器解析出来的是EndTag 标签，比如是 EndTag div，HTML 解析器会查看 Token 栈顶的元素是否是 StarTag div，如果是，就将 StartTag div 从栈中弹出，表示该 div 元素解析完成。