使用node inspect + chrome 实现Nodejs Debugger 和内存分析

笔记本:	工作杂记		
创建时间:	2018/12/18 19:28	**更新时间:**	2018/12/18 20:55
作者:	13777374547@163.com		
URL:	http://nodejs.cn/api/debugger/debugger.html		

使用node debugger + chrome 实现Nodejs Debugger 和内存分析

一，node debugger

配置

nodejs 版本： v8.11.1

Node.js 包含一个进程外的调试工具，可以通过V8检查器与内置的调试客户端访问。 要使用它，需要以 inspect 参数启动 Node.js，并带上需要调试的脚本的路径
如:

```
node inspect main.js
```

使用该方法启动Node进程后，默认会创建 127.0.0.1：9229 的套接字，用于debugger 客户端连接， 如下所示:

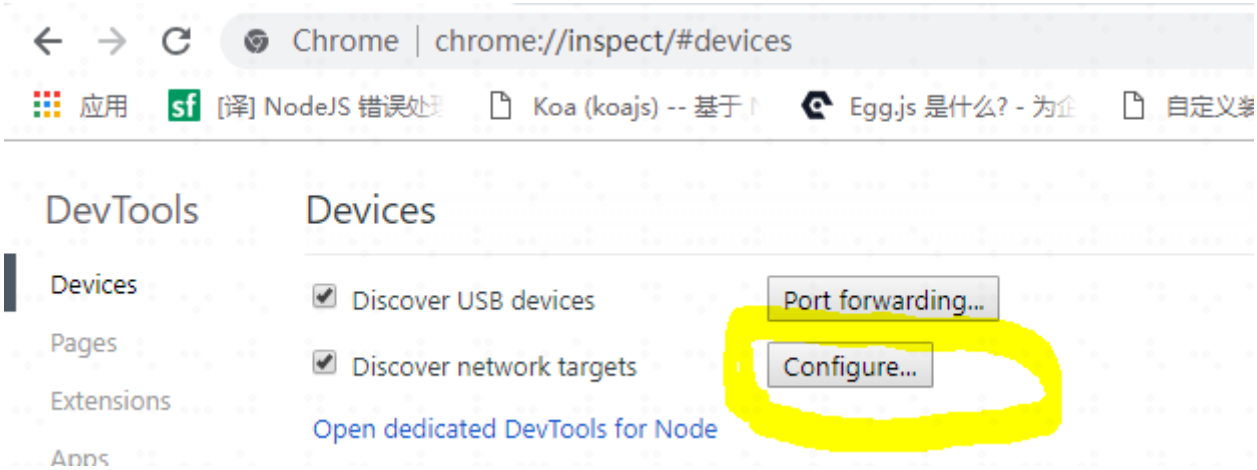
```
tcp        0      0 127.0.0.1:35497         127.0.0.1:10882         ESTABLISHED 22868/./node
```

当端口侦听在127.0.0.1 时， 所连接的debugger客户端必须与该服务在同一主机上，故可以添加参数如 ：

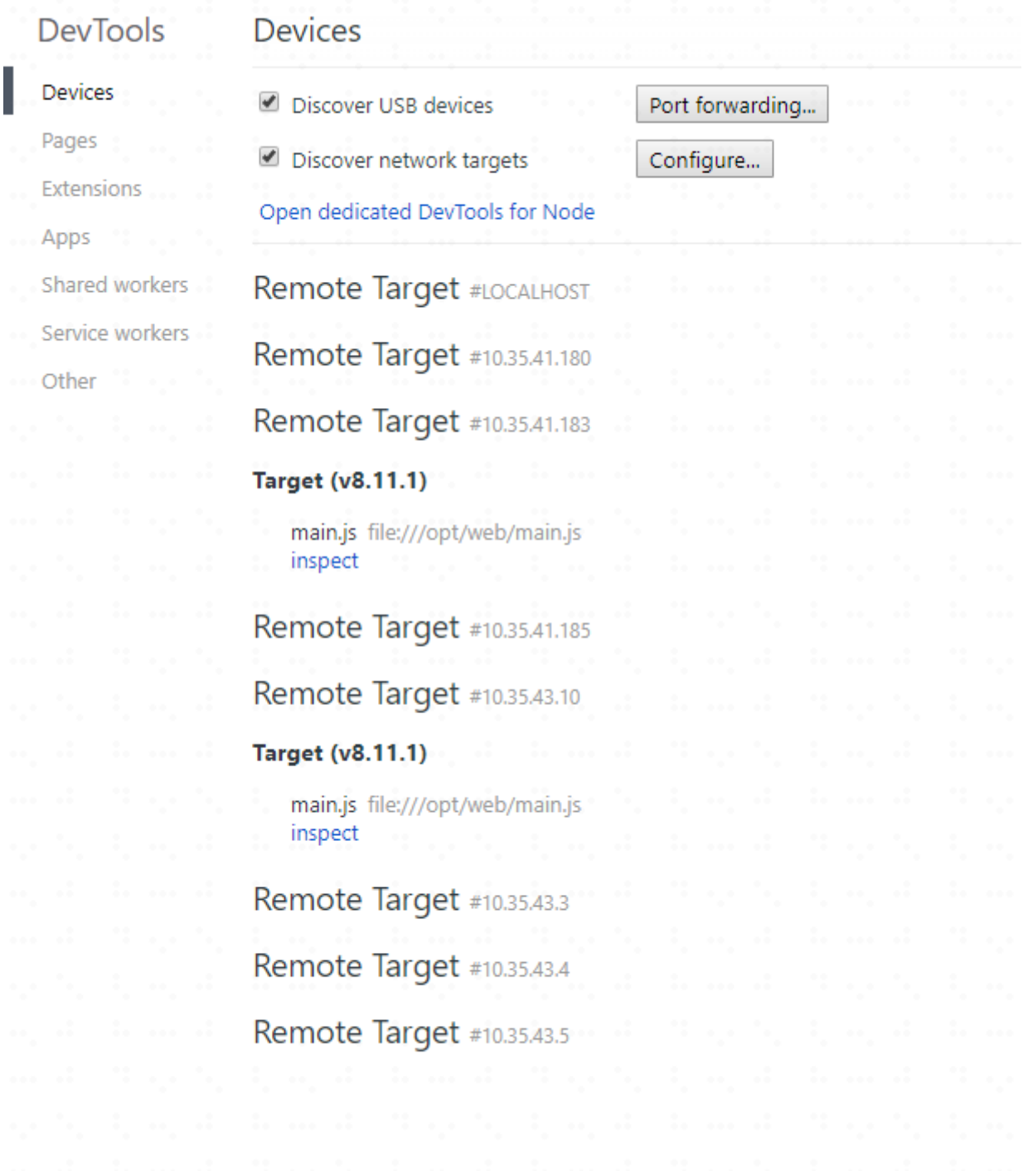
```
node --inspect=10.35.43.10:9229 main.js
```

指定主机名加端口， 此时只要客户端与服务网络通畅即可。

打开chrome （我使用的版本是69.0.3493.3， 建议使用较新版本，旧版本可能不支持该功能），在地址栏中输入chrome://inspect/#devices， 进入DevTools 配置界面



点击Configure , 弹出settings 界面， 将之前启动服务的host：port，输入即可。
随后在之前界面将会出现配置的服务侦听状态，



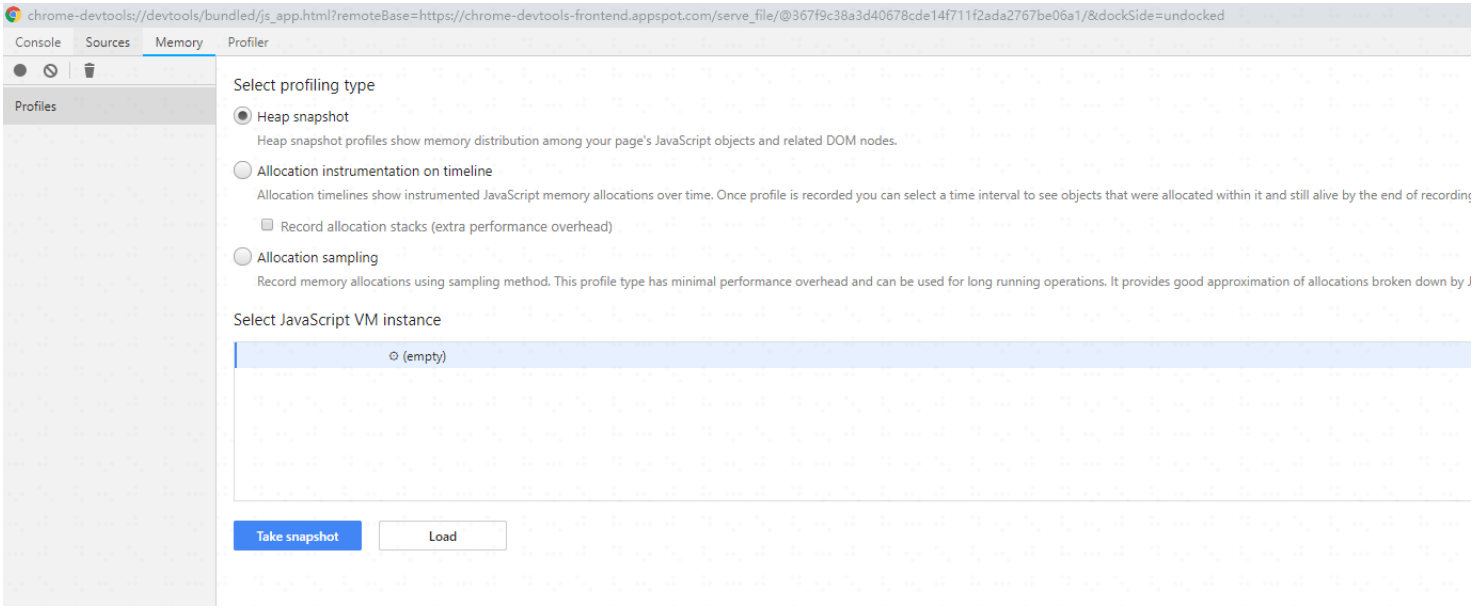
出现inspect选项，表示正在运行的， 点击inspect 开始debugger。

Debugger

打开调试工具后， Sources 选项中 可以选择当前运行的所有代码， 需要通过 在代码中插入 **debugger;** 标记，才能实现断点效果。 之后的事情就跟使用chrome 控制台调试前端代码一样简单了；

二，内存分析

还是使用debugger 那个界面， 点击Memory, 这里我们使用第一个选项 Heap snapshot， 内存快照 方式进行分析



点击Take snapshot 可以对当前的运行环境的内存堆栈生成一个快照，

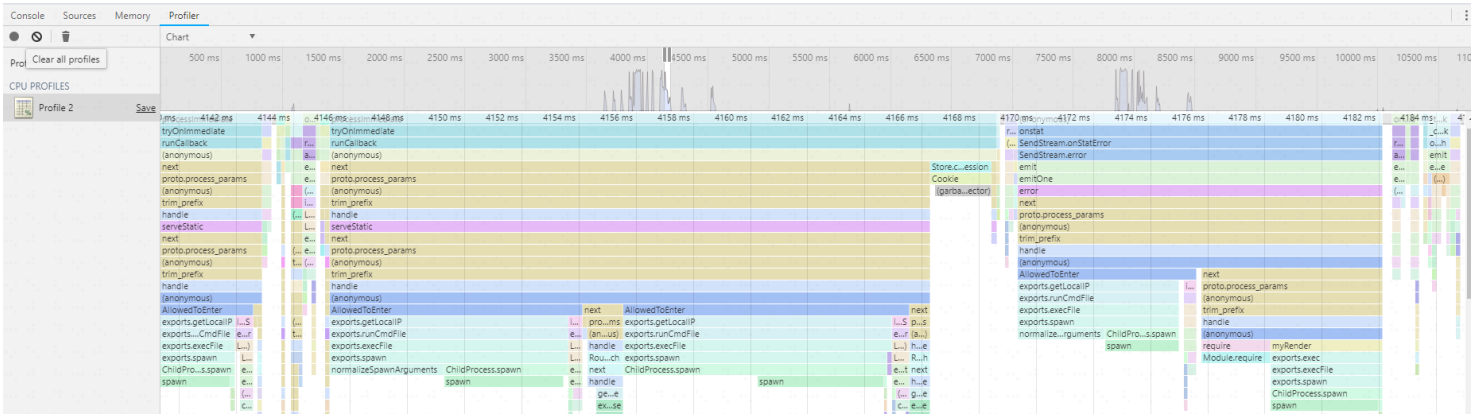
使用该工具我们可以看到 当前时刻所有的对象的内存占用， 后面几个名词解释一下

Distance：到 GC roots （GC 根对象）的距离。GC 根对象在浏览器中一般是 window 对象，在 Node.js 中是 global 对象。距离越大，说明引用越深，则有必要重点关注一下，极大可能是内存泄漏的对象。
Shallow Size：对象自身大小，不包括它引用的对象。
Retained Size：对象自身大小和它引用对象的大小，即该对象被 GC 之后所能回收的内存的大小。

一般我们会打至少三个快照进行对比分析，因为单一时刻的数据意义并不大。

三，方法耗时

点击Profile， 我们可以通过这个工具记录一段时间内的CPU耗时，从而分析函数方法中耗时较大的部分，并进行相应的优化。



其中记录这段时间内所有执行方法的耗时情况， 可以放大图表寻找，如果耗时较大，占用时间较长， 意味着需要进行更深层次的算法优化。
点击详细的方法名称，还能自动跳转到对应的代码，十分方便。

以上便是使用node 自身携带的调试器加上Chrome浏览器相结合，从而实现服务端的nodejs 在浏览器中debugger 的步骤， 对于我们排查nodejs 的异常问题，提供了方法。

Design by Jayce Mei