高级node第七单元（6.4）

2020年6月4日 9:01

在chrome开发者工具中调试node程序

1. 保证自己的node程序进程不会中断

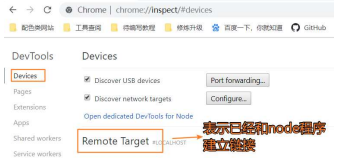
```
1
2  const events = require('events');
3
4  console.log(1);
5  console.dir(Buffer);
6  console.dir(events);
7  console.log(2)
8  console.log(3)
9
0  setTimeout(() => { }, 300000)
```

2. 在启动node程序的时候添加 --inspect选项

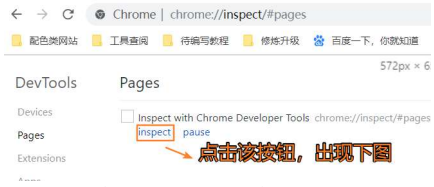
```
λ node-dev --inspect client.js
```

```
node --inspect client.js
```

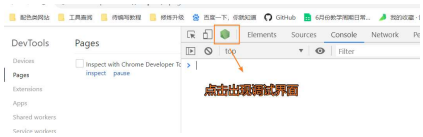
3. 在chrome浏览器的地址栏中红输入： chrome://inspect



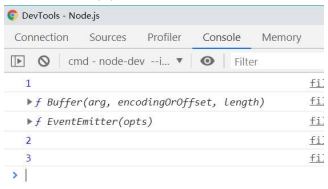
4. 点击pages选项



5. 出现如下图，点击对应按钮，出现调试界面



6. 调试界面如下图



深入理解webSocket

1. http的互动性差，不适合实时性的应用，但是可以模拟
   a. 轮询（实时性差，浪费性能）
   b. 长链接（浪费性能，应用复杂度高）
2. websocket协议的机制是浏览器和服务器建立链接后，可以
3. 随时互相通信，但是建立链接的主动方依然是浏览器，双方都可以主动断开
4. websocket协议的url地址是 ws://域名:端口/路径？查询参数#锚链接
5. websocket没有跨域

websocket的连接过程

1. 浏览器端主动发起ws链接

```
var client = new WebSocket('ws://localhost:8080');
```

2. 触发服务端的 connection事件

```
wsServer.on('connection', (client) => {
  console.log('client', client);
  // console.log('已经与浏览器建立链接了');
```

同时生成一个client对象，该对象在服务端代表了和浏览器端的通信

3. 客户端的client对象和服务端对应生成的client对象是相互呼应的，他们都具有一个readyState属性（0,1,2,3）

   0 => 链接正在建立

   1 => 链接已经建立成功，可以开始通信

   2 => 链接正在关闭

   3 => 链接已经关闭

   两个端的client对象中的readyState属性值同时开始变化，从0到1

4. 两个client对象，可以互相通过send方法给对方发送消息

   浏览器端的send发送消息，服务端箭监听message事件进行接收

```
let y = ev.clientY - disy;
client.send(JSON.stringify({ x, y }));
oBox.style.left = x + 'px';
```

```
client.on('message', (msg) => {
  // 得到某一方发送的消息后，发给另外的人
  clients.filter(item => item !== client).forEach(item => {
    console.log(msg);
    item.send(msg);
  });
})
// 监听客户端关闭
```

   服务端的send发送消息，浏览器端监听message事件进行接收

```
clients.filter(item => item !== client).forEach(item => {
  console.log(msg);
  item.send(msg);
});
```

```
client.onmessage = ({ data }) => {
  console.log(data);
  let { x, y } = JSON.parse(data);
  oBox.style.left = x + 'px';
  oBox.style.top = y + 'px';
}
```

5. 结束通信关闭链接

   a. 浏览器端断开链接（1.关闭浏览器，2.调用close方法），服务端通过close事件监听浏览器端的断开

```
// 监听客户端关闭
client.on('close', () => {
  console.log(client);
  // 筛查关闭的客户端
  clients = clients.filter(item => item.readyState === 1);
  console.log(`当前链接人数是${clients.length}`)
})
```

   b. 服务器端断开链接（1.关闭服务器，2.调用close方法），浏览器端通过close事件监听服务器的断开

```
client.onclose = () => {
  console.log(client.readyState, '链接已经关闭');
}
```

6. 在整个链接，通信，关闭的过程中，我们的readyState永远实时更新的

联机五子棋

1. 如何交互
   a. 由主人方创建房间
   b. 客人方获取房间标识进入房间
   c. 下棋的过程是又主人方和客人方进行的
   d. 服务器要实现主人方和客人方信息的交换和限制
   e. 主人和客人身份的识别

2. 服务端的实现（见以下目录文件）