爱创课堂前端培训

# NodeJS

第5天课堂笔记（本课程共6天）

班级：北京前端训练营X期

讲师：李兰波

日期：2019年1月27日

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

**蓝色是自己添加的笔记**

# WebSocket

这是一个新的前后端交互的协议。与HTTP协议同级别。

HTTP协议是一个无状态的协议。(连接一下就会断开)

WebSocket是一个有状态的协议。 (和http协议发起请求的方式是一样的 但是依赖于HTTP协议的)

使客户端和服务器端之间的数据交换变得更加简单 允许服务器主动向客户端推送数据

浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。

搭建服务器的方式现在有两种(1. 原生， 2. Express(其实也是原声书写的方式，只是优化了原声的书写方式))

# Socket.io

这是一个NodeJS的第三方模块。

它允许NodeJS服务器与前端浏览器更简便的操作WebSocket对象。

下载: npm install socket.io

Socket.io默认只与原生服务器配合：

|  |
| --- |
| 1. // 引入HTTP 2. var http = require("http"); 3. // 引入socket.io 4. var socket\_io = require("socket.io"); 5. // 搭建原生服务器 6. var server = http.createServer(); 7. // 将原生服务器放入socket\_io 8. var io = socket\_io(server); 9. // 监听端口号 10. server.listen(3000); |

此时，创建服务器，连函数都没有传递，按说应该没有任何响应。但是，当我们访问: /socket.io/socket.io.js

|  |
| --- |
|  |

（原因： socket.io统一了前端的所有的浏览器的发送webSocket的请求的方式， 统一方式，就是让前端请求该文件，该文件被引入到前端，然后才统一了）

这是一个前端JS文件。我们在前端想要使用socket.io提供的简便的处理方式时，就要引入该文件。

jQuery统一了前端所有浏览器的DOM操作

该文件统一了前端所有浏览器的发送websocket请求的方式

Socket.io无法与express创建的服务器配合使用：

|  |
| --- |
| 1. // 引入express 2. var express = require("express"); 3. // 引入socket.io 4. var socket\_io = require("socket.io"); 5. // 搭建express服务器 6. var app = express(); 7. // 将express服务器放入socket\_io 8. var io = socket\_io(app); 9. // 监听端口号 10. app.listen(3000); |

此时，会报错：

|  |
| --- |
|  |

可以将express的实例化对象转换成原生的对象即可。而且转化后的原生地形依旧保留Express的一切特点

转换：

|  |
| --- |
| 1. // 引入express 2. var express = require("express"); 3. // 因为socket.io的原因 我们需要将express搭建出来的app 转为原生的对象 4. var http = require("http"); 5. // 引入socket.io 6. var socket\_io = require("socket.io"); 7. // 搭建express服务器 8. var app = express(); 9. // 转换对象 10. var server = http.Server(app); 11. // 将express服务器放入socket\_io 12. var io = socket\_io(server); 13. // 监听端口号 14. server.listen(3000); |

表现：

|  |
| --- |
|  |

# 初始化方式

## 3.1 前端初始化

当服务器配置好之后，前端可以通过script标签将/socket.io/socket.io.js引入。

该文件会向全局中暴露一个变量： io。 它是一个函数。当它执行的时候，会发出websocket请求。

代码：

|  |
| --- |
| 前端引入  <script type="text/javascript" src="/socket.io/socket.io.js"></script>  后端初始化向全局中增加了一个io函数 该函数一旦执行 就会申请一条持久的通道   1. // 初始化 websocket连接请求 2. var socket = io(); |

netWork面板：

|  |
| --- |
|  |

此时，socket就是持久连接在前端的操作对象。之后的所有操作都通过该对象完成。

## 3.2 后端初始化

代码：

|  |
| --- |
| 1. var server = http.Server(app); 2. var io = socket\_io(server); |

io就是后端的所有socket对象的管家。

它有一个connection事件，每当有前端与服务器建立持久连接时，就会触发该事件。

|  |
| --- |
| 1. // 监听connection事件 2. io.on("connection", function(socket) { 3. // 这里的socket就是本次建立起来的持久连接在后端的对象 4. console.log("有人来了"); 5. }) |

触发一次：

|  |
| --- |
|  |

## 3.3 前后端交互

它的交互方式与观察者模式很像。

如果前端想要触发后端的事情，后端必须先监听该事件。

前端可以通过emit触发后端on的事件

后端可以通过emit触发前端on的事件

后端监听某一个事件：

|  |
| --- |
| 1. // 后端监听某一个事件 2. socket.on("hello", function() { 3. console.log("heelo"); 4. }) |

前端触发该事件：

|  |
| --- |
|  |

后端响应：

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |