

## NodeJS

- http - http模块 - require('http')
  - var http = require('http');
  - var server = http.createServer([requestListener])
    - 创建并返回一个HTTP服务器对象
    - requestListener: 监听到客户端连接的回调函数
  - server.listen(port, [hostname], [backlog], [callback])
    - 监听客户端连接请求, 只有当调用了listen方法以后, 服务器才开始工作
    - port: 监听的端口
    - hostname: 主机名 (IP/域名)
    - backlog: 连接等待队列的最大长度
    - callback: 调用listen方法并成功开启监听以后, 会触发一个listening事件, callback将作为该事件的执行函数



## NodeJS

- http - http模块 - require('http')
  - listening事件: 当server调用listen方法并成功开始监听以后触发的事件
  - error事件: 当服务开启失败的时候触发的事件
    - 参数err: 具体的错误对象
  - request事件: 当有客户端发送请求到该主机和端口的请求的时候触发
    - 参数request: http.IncomingMessage的一个实例, 通过他我们可以获取到这次请求的一些信息, 比如头信息, 数据等
    - 参数response: http.ServerResponse的一个实例, 通过他我们可以向该次请求的客户端输出返回响应



// 搭建一个http服务器, 用于处理用户发送的http请求, 需要使用node提供的一个模块

http

### // 加载一个http模块

```
var http = require('http');
```

### // 通过http模块下的createServer创建并返回一个web服务器对象

```
var server = http.createServer();
```

```
server.listen(8080, 'localhost');
```

// 不填参数系统自动分配一个端口给你用, 但是每次都会变。要自己设置一个大的端口号

// 端口就是网卡大通道的分支, 一个端口只能被一个应用程序使用

```
console.log(server.address());
```

## // 错误处理事件

```
server.on('error',function (res) {  
    console.log(res);  
});
```

## // 调用listen方法成功开始监听以后触发的事件

```
server.on('listening',function () {  
    console.log("开始监听了。。。")  
});
```

## // request事件 有客户端发来请求触发的事件

```
server.on('request',function () {  
    console.log("有客户端请求了")  
});
```