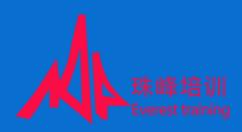
- TCP (Transmission Control Protocol 传输控制协议)是一个可靠的 面向连接的传输层协议。
- 它可以让你将数据从一台计算机完整 有序 地传输到另一台计算机,内置机制能够控制数据包的延迟率及 表包率 不会太高。
- 发送方将数据转为 字节流分成,将数据交给IP层。接收方接收后重新 装配成原始的数据
- TCP对字符和字符编码是完全 元知 的,不同的编码会导致传输的字节数不同。
- TCP使用 流控制 来确保两点之间传输数据的平衡,以防止快速的发送 方淹没慢速的接收方
- TCP在传输前要经过 3 次握手 才能形成会话,只有会话形成后,服务器端和客户端才能互相发送数据,
- 在会话过程中,服务器和客户端分别提供一个套接字,这两个套接字训

net模块用于实现TCP服务器端和客户端之间的通信。

- options 参数
 - allowHalfOpen 属性值为false时, TCP服务器收到客户端的 FIN 包时会回发 FIN 包,为true时,服务器收到客户端 FIN 包不回发 FIN 包
- connectionListener=function(socket){} 客户端 造機 时的回调函数
 - socket 表示服务器监听的 socket 端口对象
- 返回被创建的服务器server
- listen(port,[host],[backlog],[callback])
 - port 监听的 端口号
 - host 监听的 IP地址或主机名

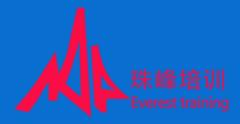


socket

- net.Socket代表一个socket端口对象
- socket端口对象可用来 读取 客户端发送的流数据,读到数据时触发

data 事件

socket.on('data',function(data){});



DJ XII DO CICC C

```
var socket = new net.Socket([options]);
```

- options
 - fd socket文件描述符

 - allowHalfOpen 是否允许半开连接
- 连接TCP服务器

```
socket.connect(port,[host],[connectListener]);
```

- port 端口
- host 主机地址
- connectListener 连接成功后的监听
- 向服务器发送数据

```
socket.write(data,[encoding],[callback]);
```



net类方法

- net.isIP 判断字符串是否是IP
- net.isIPv4 是否是IPv4地址
- net.isIPv6 是否是IPv6地址



UDP

- TCP是基于连接的协议,进行通信前客户端与服务器端要先建立连接, UDP是面向非進度的协议,可能直接发数据包。
- 不要求分组顺序到达传输层中
- 受网络影响可能 = 失 数据包
- 资源消耗少,处理速度快,适合音频,视频和普通数据传输。
- UDP 协议中的包称为数据报 datagram



创建socket

var socket = dgram.createSocket(type,[callback]);

- type **协议类型**可以为udp4或udp6
- callback = function(msg,rinfo) 收到数据时的回调函数
 - msg 收到的 数据
 - rinfo
 - address 发送者的 IP
 - family 地址 类型
 - port 发送者的socket 端口号
 - size 发送者发送的数据字节数



绑定地址和端口

socket.bind(port,[address],[callback]);

- port 绑定的端□号
- address 绑定的 IP地址
- callback 绑定后的 回调



发送数据

socket.send(buf,offset,length,port,address,[callback]);

- buf 要发送的 **数据**
- offset 从缓存区中第几个字节开始发送数据
- length 发送数据的字节数
- port 接收数据的端□号
- address 接收数据的 IP地址
- callback function(err,bytes){} 发送完毕时所调的回调函数
 - err 发送出错时的触发的错误对象
 - bytes 发送数据的字节数

