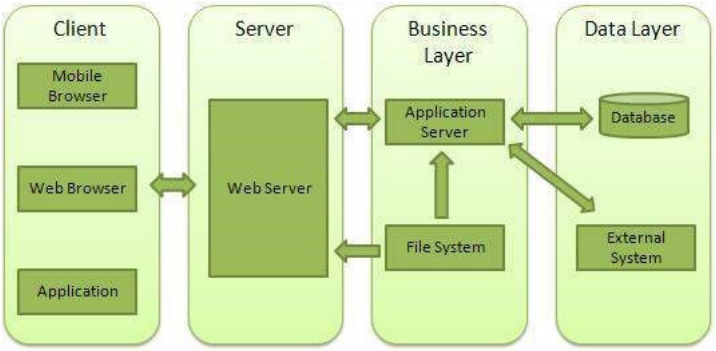


- Client - 客户端，一般指浏览器，浏览器可以通过 HTTP 协议向服务器请求数据。

- Server - 服务端，一般指 Web 服务器，可以接收客户端请求，并向客户端发送响应数据。

- Business - 业务层， 通过 Web 服务器处理应用程序，如与数据库交互，逻辑运算，调用外部程序等。

- Data - 数据层，一般由数据库组成。

net：创建基于TCP或IPC的servers和clients

```
// 创建一个服务
var server = net.createServer(function socketConnet(socket) {

    // 打印客户端的ip和端口
    console.log(`${socket.remoteAddress}:${socket.remotePort}`);

});

var port = 2080;

// 监听服务端口
server.listen(port, (err) => {
    if (err) {
        console.log('端口被占用');
        return false;
    }
    console.log(`服务端正常启动监听【${port}】端口`);
})
```

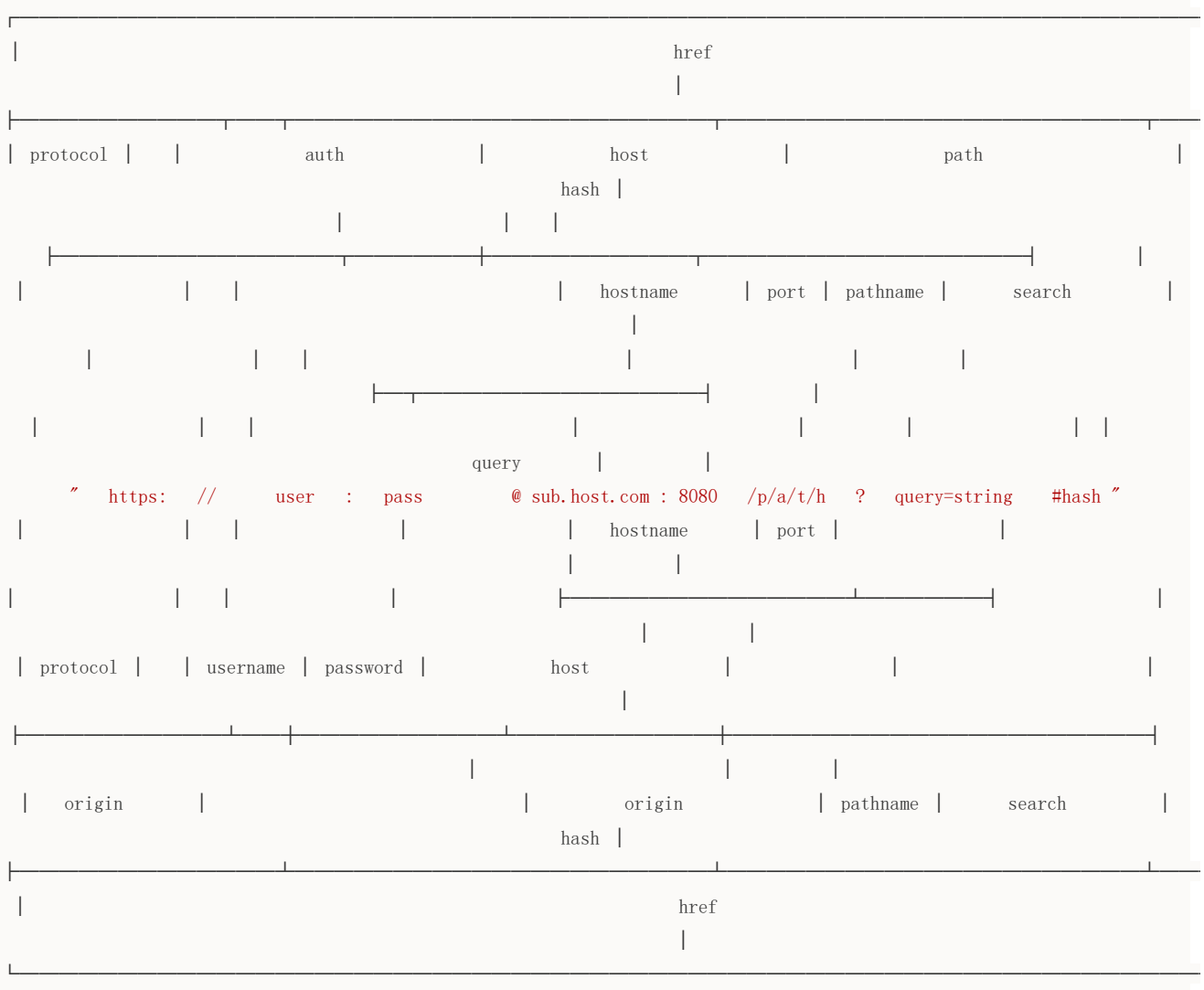
net	说明
net.createServer([options][, connectionListener])	创建一个TCP或IPC服务器
server.listen(path[, backlog][, callback])	监听服务器
socket.connect(options[, connectListener])	连接服务器

http：使用htt服务器或客服端

http	说明
http.createServer([requestListener]) server.listen(handle[, callback]) server.setTimeout([msecs][, callback]) server.close([callback])	创建一个HTTP服务器 监听http服务器 设置 socket 的超时时间。 禁止 server 接收连接
http.ServerResponse实例	HTTP服务器的response

response.writeHead(statusCode[, statusMessage][, headers]) response.write(chunk[, encoding][, callback]) **response.end([data][, encoding][, callback])** response.setTimeout(msecs, callback) response.setHeader(name, value) response.getHeader(name) response.removeHeader(name) ----- response.statusCode	发送一个响应头给请求 发送响应体 **所有的响应头和响应体已经发送；服务器可以认为消息结束。** 设置 socket 超时时间 设置默认头某个字段内容。 读取一个在队列中但是还没有被发送至客户端的header。 从即将发送的队列里移除头。 ------ 响应状态码
http.request(options[, callback])	发送HTTP请求
http.get(options[, callback])	使用GET请求，并自动调用 req.end()。

url：解析url

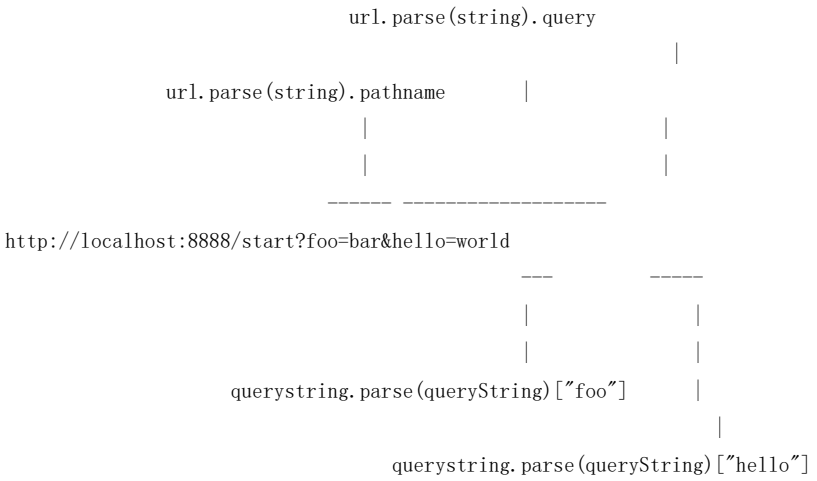


url api	说明
url.parse(urlStr[, parseQueryString][, slashesDenoteHost])	将一个 URL 字符串解析为一个 URL 对象
url.format(urlObj)	将一个 URL 对象格式化为字符串的形式
url.resolve(from, to)	用于组合 URL 成员为完整的 URL 字符串

querystring：操作查询字符串

querystring api	说明

querystring.stringify(obj[, sep[, eq[, options]]])	序列化查询字符串
querystring.parse(str[, sep[, eq[, options]]])	反序列化查询字符串
querystring.escape(str)	转义查询字符串
querystring.unescape(str)	反转义查询字符串




```
//实现路由

var http = require("http");
var url = require("url");

function start() {
   function onRequest(request, response) {
      var pathname = url.parse(request.url).pathname;
      console.log("Request for " + pathname + " received.");
      response.writeHead(200, {"Content-Type": "text/plain"});
      response.write("Hello World");
      response.end();
   }

   http.createServer(onRequest).listen(8888);
   console.log("Server has started.");}

exports.start = start;
```