Node.js EventEmitter Node.js 路由

# Node.js 函数

在JavaScript中，一个函数可以作为另一个函数的参数。我们可以先定义一个函数，然后传递，也可以在传递参数的地方直接定义函数。

Node.js中函数的使用与Javascript类似，举例来说，你可以这样做：

```
function say(word) {
  console.log(word);
}

function execute(someFunction, value) {
  someFunction(value);
}

execute(say, "Hello");
```

以上代码中，我们把 say 函数作为execute函数的第一个变量进行了传递。这里传递的不是 say 的返回值，而是 say 本身！

这样一来， say 就变成了execute 中的本地变量 someFunction ，execute可以通过调用 someFunction()（带括号的形式）来使用 say 函数。

当然，因为 say 有一个变量， execute 在调用 someFunction 时可以传递这样一个变量。

---

## 匿名函数

我们可以把一个函数作为变量传递。但是我们不一定要绕这个"先定义，再传递"的圈子，我们可以直接在另一个函数的括号中定义和传递这个函数：

```
function execute(someFunction, value) {
  someFunction(value);
}

execute(function(word){ console.log(word) }, "Hello");
```

我们在 execute 接受第一个参数的地方直接定义了我们准备传递给 execute 的函数。

用这种方式，我们甚至不用给这个函数起名字，这也是为什么它被叫做匿名函数 。

---

## 函数传递是如何让HTTP服务器工作的

带着这些知识，我们再来看看我们简约而不简单的HTTP服务器：

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

```
http.createServer(function(request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("Hello World");
  response.end();
}).listen(8888);
```

现在它看上去应该清晰了很多：我们向 createServer 函数传递了一个匿名函数。

用这样的代码也可以达到同样的目的：

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

function onRequest(request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("Hello World");
  response.end();
}

http.createServer(onRequest).listen(8888);
```

← Node.js EventEmitter

Node.js 路由 →