**EventEmitter 类**

events 模块只提供了一个对象： events.EventEmitter。

EventEmitter 的核心就是事件触发与事件监听器功能的封装。

通过require("events");来访问该模块。

// 引入 events 模块

var events = require('events');

// 创建 eventEmitter 对象

var eventEmitter = new events.EventEmitter();

下面我们用一个简单的例子说明 EventEmitter 的用法：

//event.js 文件

var EventEmitter = require('events').EventEmitter;

var event = new EventEmitter();

event.on('some\_event', function() {

    console.log('some\_event 事件触发');

});

setTimeout(function() {

    event.emit('some\_event');

}, 1000);

运行这段代码，1 秒后控制台输出了 **'some\_event 事件触发'**。

$ node event.js

some\_event 事件触发

EventEmitter 的每个事件由一个事件名和若干个参数组成

让我们以下面的例子解释这个过程：

//event.js 文件

var events = require('events');

var emitter = new events.EventEmitter();

emitter.on('someEvent', function(arg1, arg2) {

    console.log('listener1', arg1, arg2);

});

emitter.on('someEvent', function(arg1, arg2) {

    console.log('listener2', arg1, arg2);

});

emitter.emit('someEvent', 'arg1 参数', 'arg2 参数');

执行以上代码，运行的结果如下：

$ node event.js

listener1 arg1 参数 arg2 参数

listener2 arg1 参数 arg2 参数

接下来我们来具体看下 EventEmitter 的属性介绍。

**方法**

|  |  |
| --- | --- |
| **序号** | **方法 & 描述** |
| 1 | **addListener(event, listener)** 为指定事件添加一个监听器到监听器数组的尾部。 |
| 2 | **on(event, listener)** 为指定事件注册一个监听器，接受一个字符串 event 和一个回调函数。  server.on('connection', function (stream) {  console.log('someone connected!');  }); |
| 3 | **once(event, listener)** 为指定事件注册一个单次监听器，即 监听器最多只会触发一次，触发后立刻解除该监听器。  server.once('connection', function (stream) {  console.log('Ah, we have our first user!');  }); |
| 4 | **removeListener(event, listener)**  移除指定事件的某个监听器，监听器必须是该事件已经注册过的监听器。  它接受两个参数，第一个是事件名称，第二个是回调函数名称。  var callback = function(stream) {  console.log('someone connected!');  };  server.on('connection', callback);  // ...  server.removeListener('connection', callback); |
| 5 | **removeAllListeners([event])** 移除所有事件的所有监听器， 如果指定事件，则移除指定事件的所有监听器。 |
| 6 | **setMaxListeners(n)** 默认情况下， EventEmitters 如果你添加的监听器超过 10 个就会输出警告信息。 setMaxListeners 函数用于提高监听器的默认限制的数量。 |
| 7 | **listeners(event)** 返回指定事件的监听器数组。 |
| 8 | **emit(event, [arg1], [arg2], [...])** 按监听器的顺序执行执行每个监听器，如果事件有注册监听返回 true，否则返回 false。 |

**error 事件**

EventEmitter 定义了一个特殊的事件 error

当 error 被触发时，EventEmitter 规定如果没有响 应的监听器，Node.js 会把它当作异常，退出程序并输出错误信息。

我们一般要为会触发 error 事件的对象设置监听器，避免遇到错误后整个程序崩溃。例如：

var events = require('events');

var emitter = new events.EventEmitter();

emitter.emit('error');

运行时会显示以下错误：

node.js:201

throw e; // process.nextTick error, or 'error' event on first tick

^

Error: Uncaught, unspecified 'error' event.

at EventEmitter.emit (events.js:50:15)

at Object.<anonymous> (/home/byvoid/error.js:5:9)

at Module.\_compile (module.js:441:26)

at Object..js (module.js:459:10)

at Module.load (module.js:348:31)

at Function.\_load (module.js:308:12)

at Array.0 (module.js:479:10)

at EventEmitter.\_tickCallback (node.js:192:40)

**继承 EventEmitter**

大多数时候我们不会直接使用 EventEmitter，而是在对象中继承它。包括 fs、net、 http 在内的，只要是支持事件响应的核心模块都是 EventEmitter 的子类。

为什么要这样做呢？原因有两点：

首先，具有某个实体功能的对象实现事件符合语义， 事件的监听和发生应该是一个对象的方法。

其次 JavaScript 的对象机制是基于原型的，支持 部分多重继承，继承 EventEmitter 不会打乱对象原有的继承关系。