Scaling Your Node Application

Paul O'Fallon
@paulofallon

pluralsight
hardcore developer training

Outline

- Creating child processes in Node
- Scaling your Node app with the “cluster” module

Creating Child Processes

The "child_process" module provides several ways to invoke a process:

1. **spawn(command, [args], [options])**
   - Launches a new process with "command" and "args"
   - Returns a ChildProcess object that…
     - is an EventEmitter and emits "exit", "close" and "disconnect" events
     - has streams for stdin, stdout and stderr that can be piped to/from

2. **exec(command, [options], callback)**
   - Runs "command" string in a shell
   - Callback is invoked on process completion with error, stdout, stderr

3. **execFile(file, [args], [options], callback)**
   - Similar to "exec", except "file" is executed directly, rather than in a subshell

fork()'ing additional Node processes

There is one more way to invoke a child process in Node:

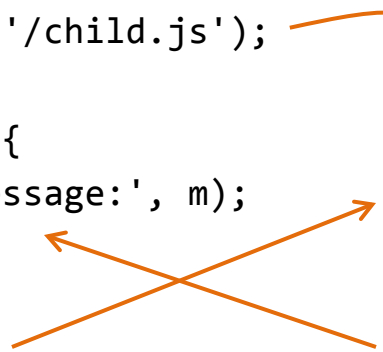
4. **fork(modulePath, [args], [options])**
   - A special version of "spawn" especially for creating Node processes
   - Adds a "send" function and "message" event to ChildProcess

parent.js

```
var cp = require('child_process');

var n = cp.fork(__dirname + '/child.js');

n.on('message', function(m) {
  console.log('PARENT got message:', m);
});

n.send({ hello: 'world' });
```

child.js

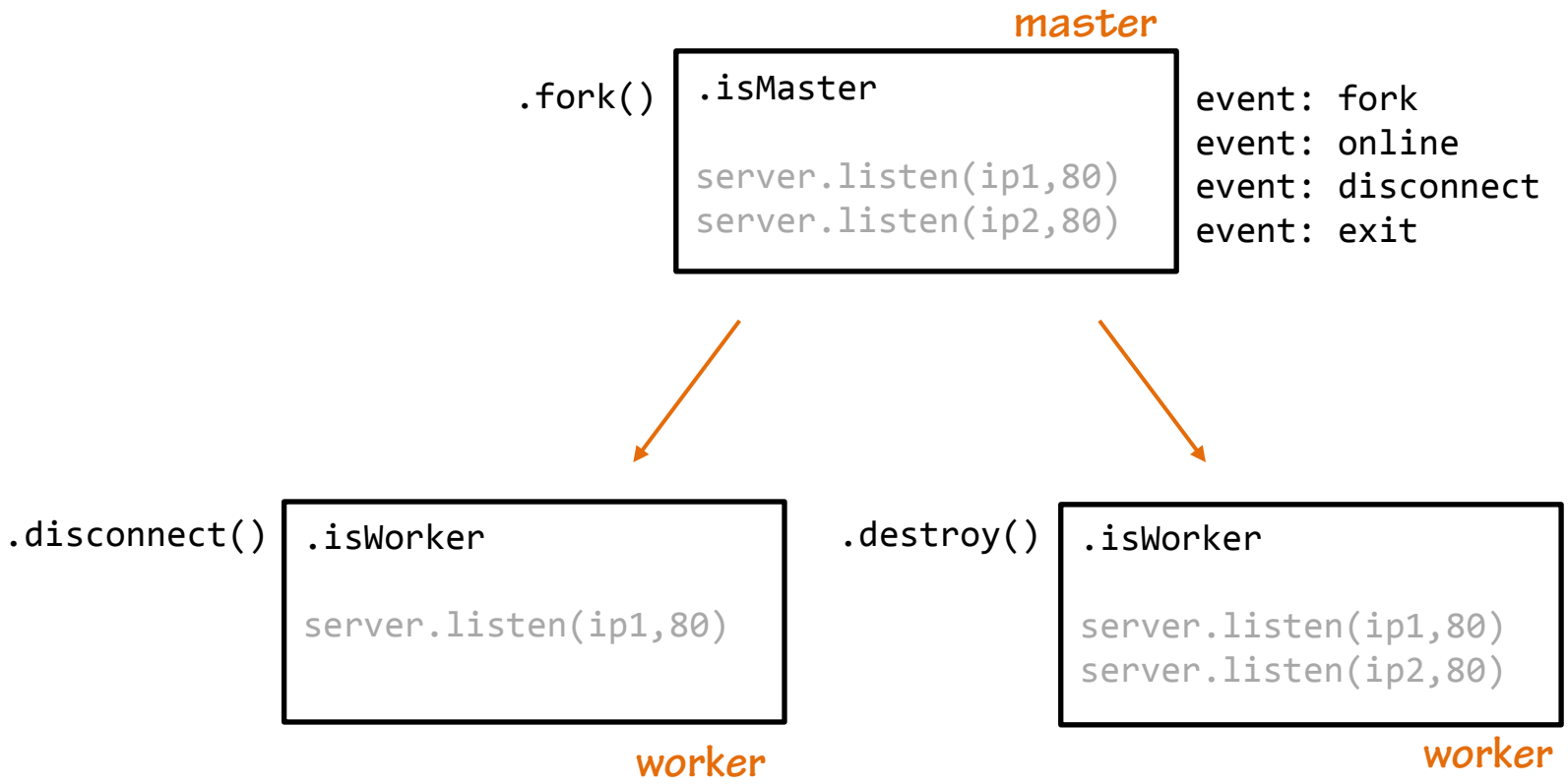
```
process.on('message', function(m) {
  console.log('CHILD got message:', m);
});

process.send({ foo: 'bar' });
```




pluralsight
hardcore developer training

Scaling with Node's "cluster" module

- An experimental module leveraging child_process.fork()
- Introduces a "Worker" class as well as master functions and events




pluralsight
hardcore developer training

Conclusion

- Creating child processes in Node
- Scaling your Node app with the “cluster” module


pluralsight
hardcore developer training

References

- Node.js documentation
  http://nodejs.org/api/