

`{"a":2,"b":3}`

until you have to give your messages types

```
{"a":2,"b":3,"type":"foo"}
```

everybody writes this at some point

```
switch (msg.type) {  
    case "foo"  
    :  
    :  
    :  
}
```

ad-hoc callback routing

function foo(a, b, cb) {
 ...
}



look ma, no switches!

```
var dnode = require('dnode');  
  
var server = dnode({  
  zing : function (n, cb) { cb(n * 100) }  
});  
server.listen(5000);
```

just call functions on the other side

```
var dnode = require('dnode');

dnode.connect(5000, function (remote) {
  remote.zing(66, function (n) {
    console.log('n = ' + n);
  });
});
```

client and server side by side

```
var dnode = require('dnode');


var server = dnode({
  zing : function (n, cb) { cb(n * 100) }
});
server.listen(5000);
```

```
var dnode = require('dnode');

dnode.connect(5000, function (remote) {
  remote.zing(66, function (n) {
    console.log('n = ' + n);
  });
});
```


pass along n...

```
var dnode = require('dnode');  
  
var server = dnode({  
  zing : function (n, cb) { cb(n * 100) }  
});  
server.listen(5000);
```

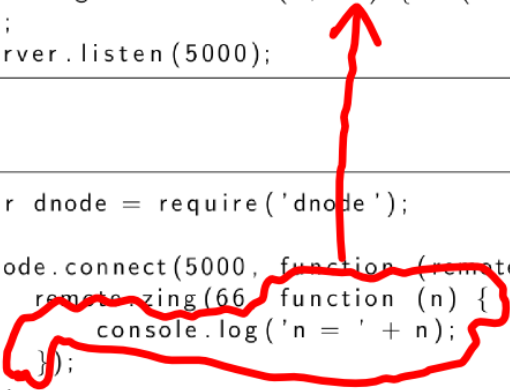


```
var dnode = require('dnode');  
  
dnode.connect(5000, function (remote) {  
  remote.zing(66, function (n) {  
    console.log('n = ' + n);  
  });  
});
```

pass along the cb...

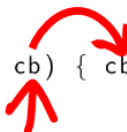
```
var dnode = require('dnode');  
  
var server = dnode({  
  zing : function (n, cb) { cb(n * 100) }  
});  
server.listen(5000);
```

```
var dnode = require('dnode');  
  
dnode.connect(5000, function (remote) {  
  remote.zing(66, function (n) {  
    console.log('n = ' + n);  
  });  
});
```

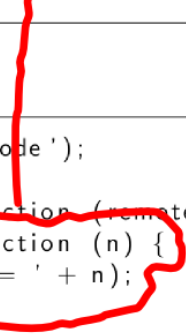


then just call the cb...

```
var dnode = require('dnode');  
  
var server = dnode({  
  zing : function (n, cb) { cb(n * 100) }  
});  
server.listen(5000);
```

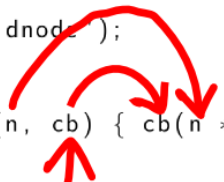


```
var dnode = require('dnode');  
  
dnode.connect(5000, function (remote) {  
  remote.zing(66, function (n) {  
    console.log('n = ' + n);  
  });  
});
```



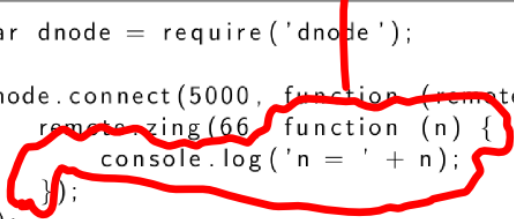
with the parameters you passed in...

```
var dnode = require('dnode');  
  
var server = dnode({  
  zing : function (n, cb) { cb(n * 100) }  
});  
server.listen(5000);
```



Red arrows indicate the flow of arguments: one arrow points from the `cb` parameter in the `zing` function to the `cb(n * 100)` call, and another arrow points from the `zing` function to the `dnode` constructor call.

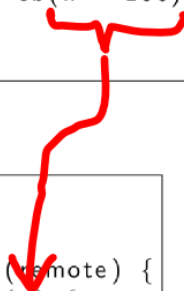
```
var dnode = require('dnode');  
  
dnode.connect(5000, function (remote) {  
  remote.zing(66, function (n) {  
    console.log('n = ' + n);  
  });  
});
```



Red arrows indicate the flow of arguments: one arrow points from the `function (n)` parameter in the nested `zing` function to the `console.log` call, and another arrow points from the `remote.zing(66, ...)` call to the `dnode.connect` constructor call.

and your callback gets the result...

```
var dnode = require('dnode');  
  
var server = dnode({  
  zing : function (n, cb) { cb(n * 100) }  
});  
server.listen(5000);
```

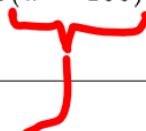


```
var dnode = require('dnode');  
  
dnode.connect(5000, function (remote) {  
  remote.zing(66, function (n) {  
    console.log('n = ' + n);  
  });  
});
```

which you can use for whatever

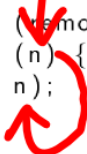
```
var dnode = require('dnode');

var server = dnode({
  zing : function (n, cb) { cb(n * 100) }
});
server.listen(5000);
```

A red bracket is drawn under the callback function `cb(n * 100)` in the first code block. A red arrow originates from the bracket and points downwards towards the second code block.

```
var dnode = require('dnode');

dnode.connect(5000, function (remote) {
  remote.zing(66, function (n) {
    console.log('n = ' + n);
  });
});
```

A red arrow points from the `remote` parameter in the `dnode.connect` call of the second code block to the `remote` parameter in the `remote.zing` call. A red bracket is drawn under the `remote.zing` call.

that's all it takes!

```
code $ node zing_server.js &  
[1] 21671  
code $ node zing_client.js  
n = 6600  
^C
```

fuck yeah, callbacks!



fuck yeah, callbacks!



Now let's do that in the browser.

web server

```
var express = require('express');
var app = express.createServer();
app.use(express.static(__dirname));
app.listen(8080);

var dnode = require('dnode');

var server = dnode({
  zing : function (n, cb) { cb(n * 100) }
});
server.listen(app);
```

network and web server

```
var express = require('express');
var app = express.createServer();
app.use(express.static(__dirname));
app.listen(8080);

var dnode = require('dnode');

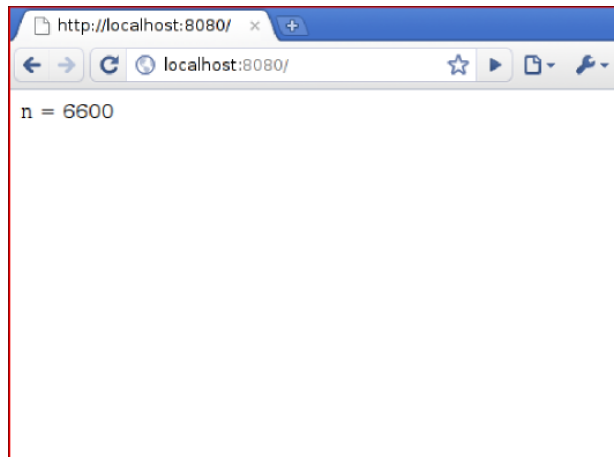
var server = dnode({
  zing : function (n, cb) { cb(n * 100) }
});
server.listen(app);
server.listen(5000);
```

just hack up an index.html:

```
<html>
<head>
  <script type="text/javascript" src="/dnode.js">
  </script>
  <script type="text/javascript">
    DNode.connect(function (remote) {
      remote.zing(66, function (n) {
        document.getElementById("res").innerHTML
          = 'n = ' + n;
      });
    });
  </script>
</head>
<body>
  <div id="res"></div>
</body>
</html>
```

zing web output

it works!

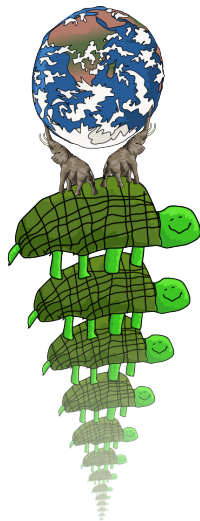


callbacks fo' real



```
remote.turtles(function (f) {  
  f(function (g) {  
    g(function (h) {  
      h('fuck yeah callbacks ');  
    });  
  });  
});
```

callbacks fo' real



```
remote.turtles(function (f) {  
  f(function (g) {  
    g(function (h) {  
      h('fuck yeah callbacks ');  
    });  
  });  
});
```

It's callbacks all the way
down.

github.com/substack/dnode-ruby

```
require 'rubygems'
require 'dnode'

DNode.new({}).connect(5050) do |remote|
  remote.f(30000) { |x| puts "x=<#{x}>" }
end
```

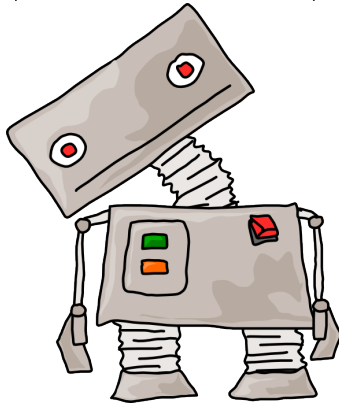

github.com/substack/dnode-perl

```
#!/usr/bin/env perl
use warnings;
use strict;
use DNode;

DNode->new({})->connect(5050, sub {
    my $remote = shift;
    $remote->{f}(1337, sub {
        my $x = shift;
        print "x = $x\n";
    })
});
```

dnode in java?!

github.com/aslakhellesoy/dnode-java



github.com/substack/dnode

