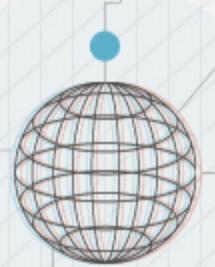


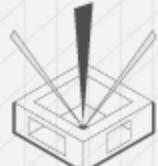


# LETS CREATE AN ECHO SERVER



```
http.createServer(function(request, response) {  
  response.writeHead(200);  
  request.on('data', function(chunk) {  
    response.write(chunk);  
  });  
  
  request.on('end', function() {  
    response.end();  
  });  
}).listen(8080)
```

A red arrow points from the line `request.pipe(response);` to the `request` object in the `request.on('end', ...)` event handler.

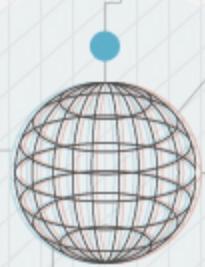


STREAMS





# LETS CREATE AN ECHO SERVER!



```
http.createServer(function(request, response) {  
  response.writeHead(200);  
  request.pipe(response);  
}).listen(8080)
```



```
$ curl -d 'hello' http://localhost:8080
```

----> Hello *on client*

*Kinda like on the command line*

```
cat 'bleh.txt' | grep 'something'
```

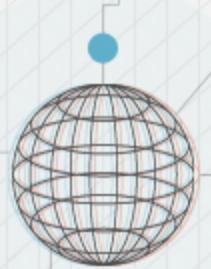


STREAMS

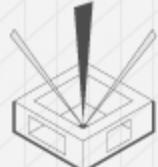




# READING AND WRITING A FILE



```
var fs = require('fs'); // require filesystem module  
  
var file = fs.createReadStream("readme.md");  
var newFile = fs.createWriteStream("readme_copy.md");  
  
file.pipe(newFile);
```

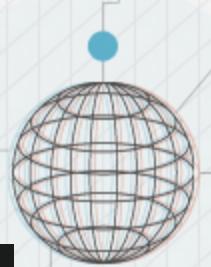


STREAMS





# UPLOAD A FILE



```
var fs = require('fs');
var http = require('http');

http.createServer(function(request, response) {
  var newFile = fs.createWriteStream("readme_copy.md");
  request.pipe(newFile);

  request.on('end', function() {
    response.end('uploaded!');
  });
}).listen(8080);
```

```
$ curl --upload-file readme.md http://localhost:8080
```

→ uploaded!



STREAMS

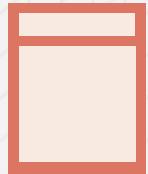




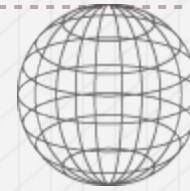
# THE AWESOME STREAMING



client



original file



server



storage



transferred file



0s

5s

10s

non-blocking

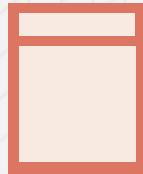




# BACK PRESSURE!



client

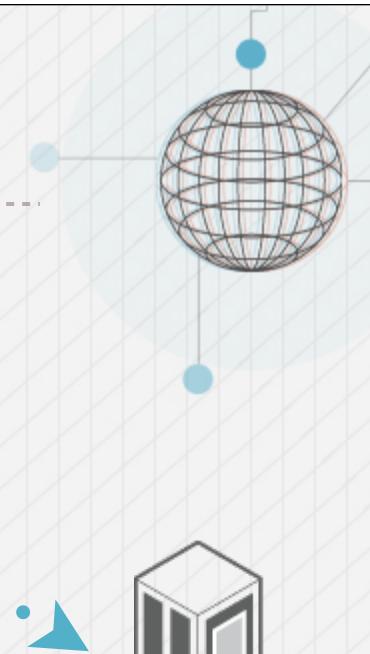


original file

✗ Writable stream slower  
than readable stream



server



storage

transferred file



Using pipe solves this problem



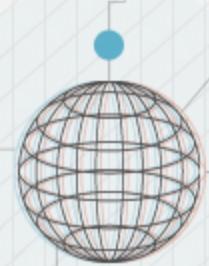
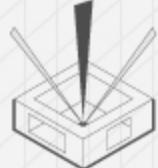


# THINK OF A MILK JUG

`milkStream.pause();`

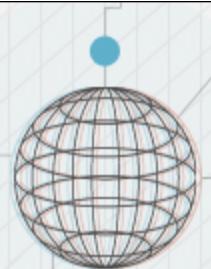


Once milk jug is drained  
`milkStream.resume();`





# PIPE SOLVES BACKPRESSURE



Pause when writeStream is full

```
readStream.on('data', function(chunk) {  
  var buffer_good = writeStream.write(chunk);  
  if (!buffer_good) readStream.pause();  
});
```

*returns false  
if kernel buffer full*

Resume when ready to write again

```
writeStream.on('drain', function(){  
  readStream.resume();  
});
```

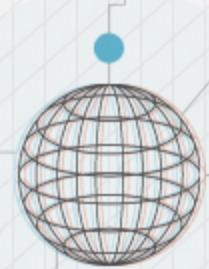
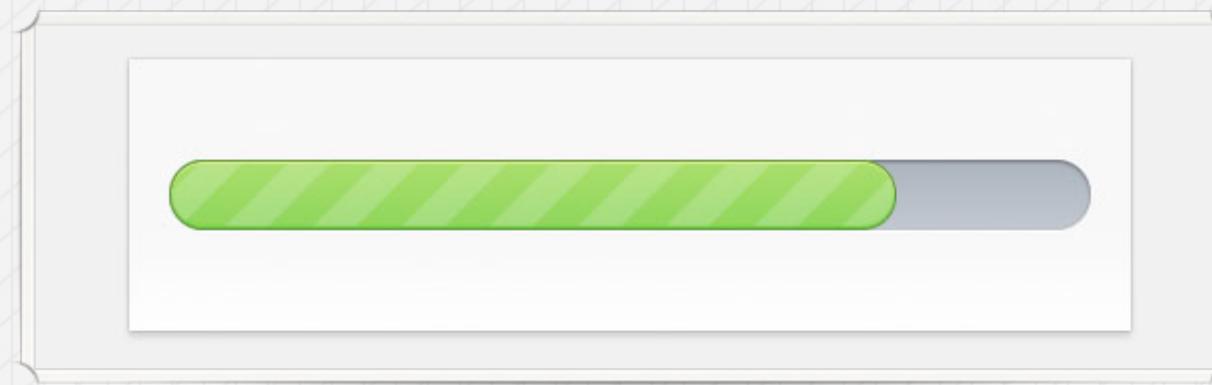
All encapsulated in

```
readStream.pipe(writeStream);
```





# FILE UPLOADING PROGRESS

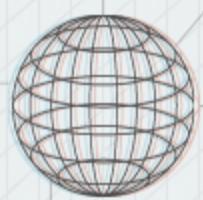


STREAMS





# FILE UPLOADING PROGRESS



```
$ curl --upload-file file.jpg http://localhost:8080
```

*Outputs:*

```
progress: 3%
progress: 6%
progress: 9%
progress: 12%
progress: 13%
...
progress: 99%
progress: 100%
```

**Choose File** No file chosen

**Upload**

*We're going to need:*

- **HTTP Server**
- **File System**



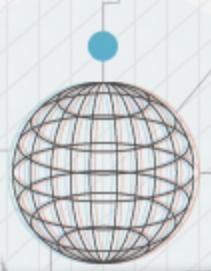
STREAMS





# DOCUMENTATION

<http://nodejs.org/api/>



*Stability Scores*

## File System

Stability: 3 - Stable

File I/O is provided by simple wrappers around standard PO `require('fs')`. All the methods have asynchronous and syn

The asynchronous form always take a completion callback as completion callback depend on the method, but the first argument operation was completed successfully, then the first argument

When using the synchronous form any exceptions are immediately exceptions or allow them to bubble up.

Here is an example of the asynchronous version:

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

fs.unlink('/tmp/hello', function (err) {
```

## Stream

Stability: 2 - Unstable

A stream is an abstract interface implemented by various objects in Node server is a stream, as is stdout. Streams are readable, writable, or both. All EventEmitters.

You can load up the Stream base class by doing `require('stream')`.

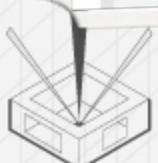
## Readable Stream

A `Readable Stream` has the following methods, members, and events.

### Event: 'data'

```
function (data) {}
```

The `'data'` event emits either a `Buffer` (by default) or a `string` if `set`



STREAMS



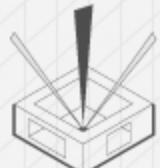
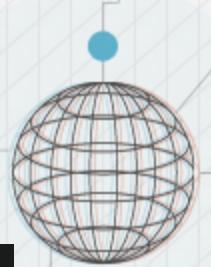


# REMEMBER THIS CODE?

```
var fs = require('fs');
var http = require('http');

http.createServer(function(request, response) {
  var newFile = fs.createWriteStream("readme_copy.md");
  request.pipe(newFile);

  request.on('end', function() {
    response.end('uploaded!');
  });
}).listen(8080);
```

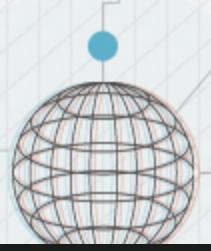


STREAMS





# REMEMBER THIS CODE?



```
http.createServer(function(request, response) {  
  var newFile = fs.createWriteStream("readme_copy.md");  
  var fileBytes = request.headers['content-length'];  
  var uploadedBytes = 0;  
  
  request.pipe(newFile);  
  
  request.on('data', function(chunk) {  
    uploadedBytes += chunk.length;  
    var progress = (uploadedBytes / fileBytes) * 100;  
    response.write("progress: " + parseInt(progress, 10) + "%\n");  
  });  
  ...  
}).listen(8080);
```

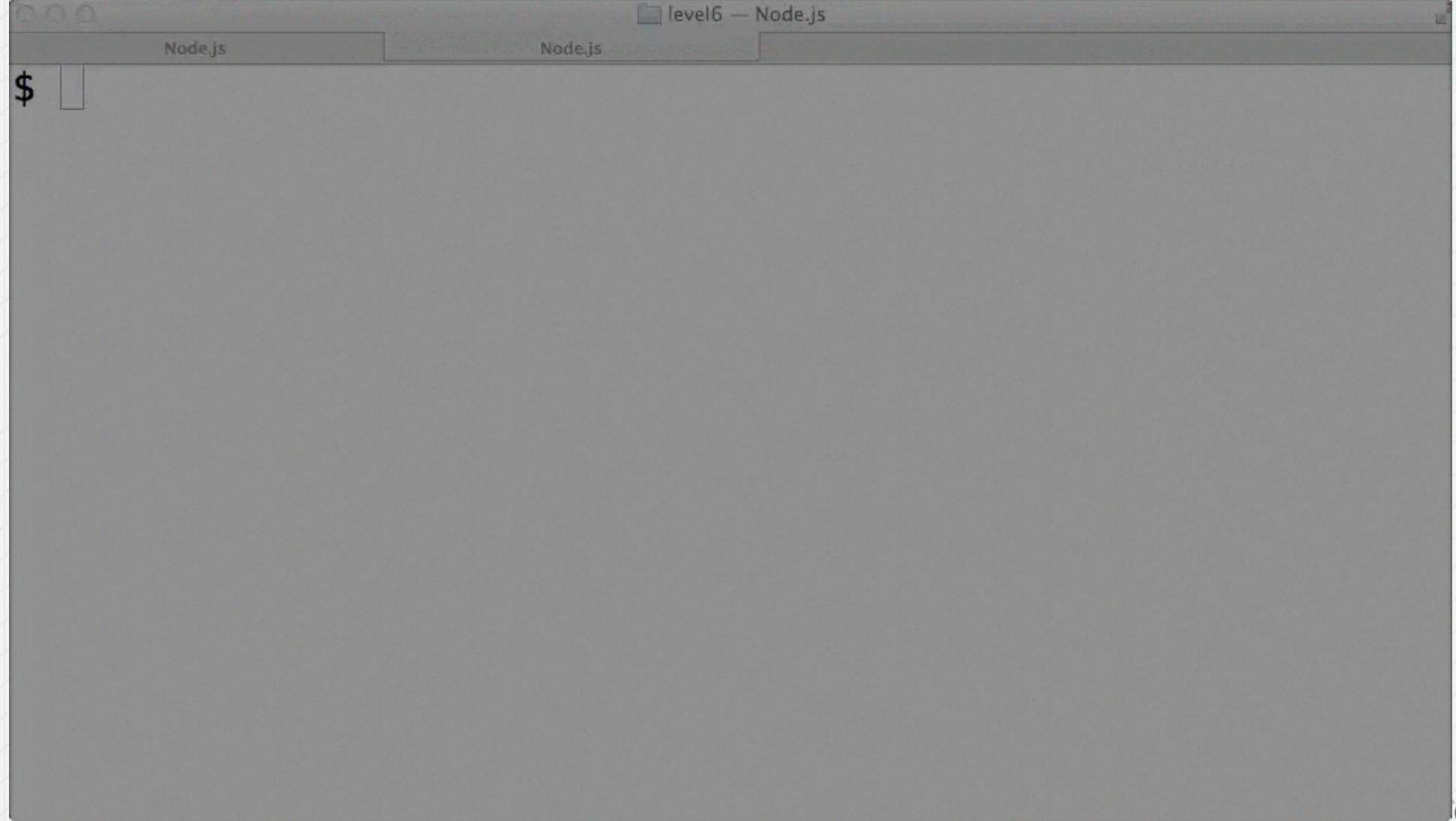


STREAMS





# SHOWING PROGRESS

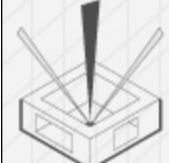
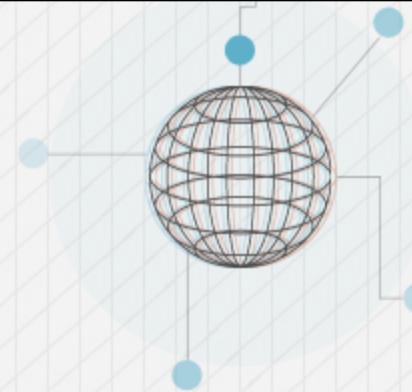


STREAMS



# MODULES

- LEVEL FOUR -





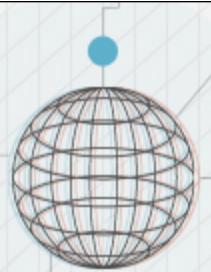
# REQUIRING MODULES

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

-----> http.js

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

-----> fs.js



How does 'require' return the libraries?

How does it find these files?

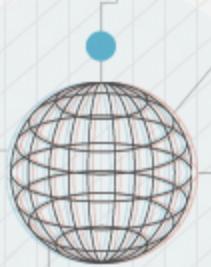


MODULES





# LETS CREATE OUR OWN MODULE



## custom\_hello.js

```
var hello = function() {  
  console.log("hello!");  
}  
  
exports = hello;
```

## custom\_goodbye.js

```
exports.goodbye = function() {  
  console.log("bye!");  
}
```

## app.js

```
var hello = require('./custom_hello');  
  
var gb = require('./custom_goodbye');  
  
hello();  
  
gb.goodbye();
```

```
require('./custom_goodbye').goodbye();
```

*If we only need to call once*

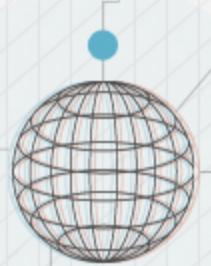


MODULES





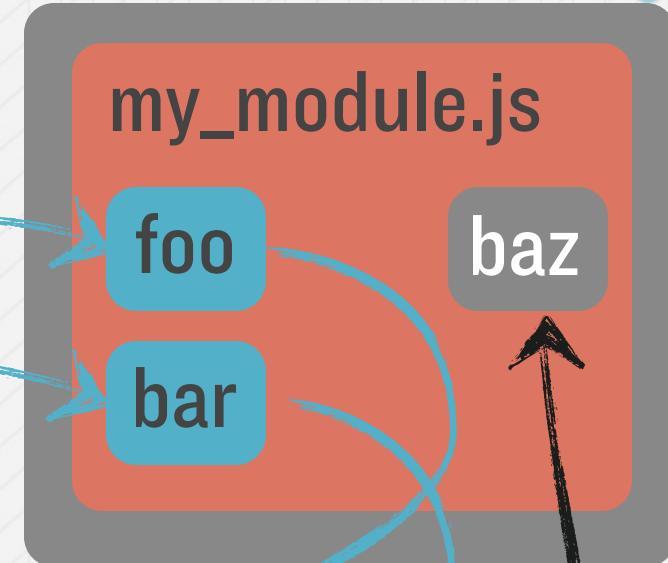
# EXPORT MULTIPLE FUNCTIONS



## my\_module.js

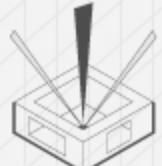
```
var foo = function() { ... }  
var bar = function() { ... }  
var baz = function() { ... }
```

```
exports.foo = foo  
exports.bar = bar
```



## app.js

```
var myMod = require('./my_module');  
myMod.foo();  
myMod.bar();
```

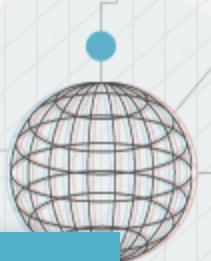


MODULES





# MAKING HTTP REQUESTS



```
var http = require('http');  
  
var message = "Here's looking at you, kid.";  
var options = {  
  host: 'localhost', port: 8080, path: '/', method: 'POST'  
}  
  
var request = http.request(options, function(response){  
  response.on('data', function(data){  
    console.log(data); logs response body  
  });  
});  
request.write(message); begins request  
request.end(); finishes request
```

app.js



MODULES





# ENCAPSULATING THE FUNCTION



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

app.js

```
var makeRequest = function(message){  
  var options = {  
    host: 'localhost', port: 8080, path: '/', method: 'POST'  
  }
```

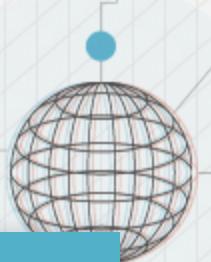
```
var request = http.request(options, function(response){  
  response.on('data', function(data){  
    console.log(data);  
  });  
});  
request.write(message);  
request.end();  
}  
  
makeRequest("Here's looking at you, kid.");
```

MODULES





# CREATING & USING A MODULE



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

make\_request.js

```
var makeRequest = function(message) {
```

```
    ...
```

```
}
```

```
exports = makeRequest;
```

```
var makeRequest = require('./make_request');
```

app.js

```
makeRequest("Here's looking at you, kid");
```

```
makeRequest("Hello, this is dog");
```

## Where does require look for modules?

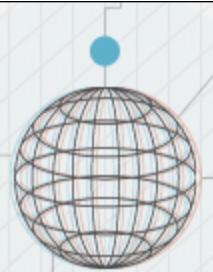


MODULES





# REQUIRE SEARCH



```
var make_request = require('./make_request')look in same directory
var make_request = require('../make_request')look in parent directory
var make_request = require('/Users/eric/nodes/make_request')
```

/Home/eric/my\_app/app.js

*Search in node\_modules directories*

```
var make_request = require('make_request')
```

- /Home/eric/my\_app/node\_modules/
- /Home/eric/node\_modules/make\_request.js
- /Home/node\_modules/make\_request.js
- /node\_modules/make\_request.js

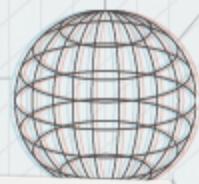


MODULES





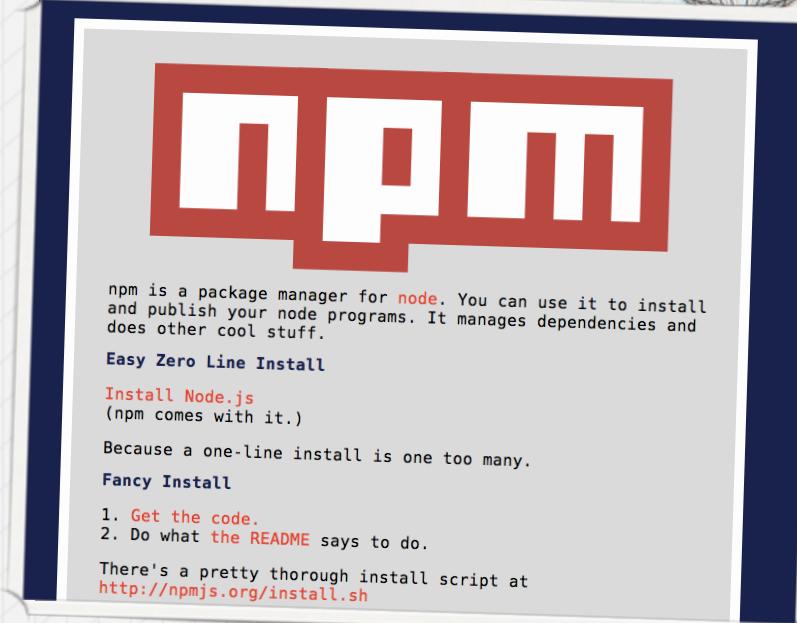
# NPM: THE USERLAND SEA



## Package manager for node

- Comes with node
- Module Repository
- Dependency Management
- Easily publish modules
- “Local Only”

“Core” is small. “Userland” is large.

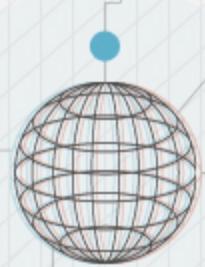


MODULES





# INSTALLING A NPM MODULE



In /Home/my\_app

```
$ npm install request
```

- - - -> <https://github.com/mikeal/request>

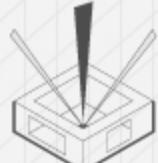
Installs into local node\_modules directory

Home / my\_app / node\_modules / request

In /Home/my\_app/app.js

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

Loads from local node\_modules directory



MODULES





# LOCAL VS GLOBAL

Install modules with executables globally

```
$ npm install coffee-script -g global
```

```
$ coffee app.coffee
```

Global npm modules can't be required

```
var coffee = require('coffee-script');
```

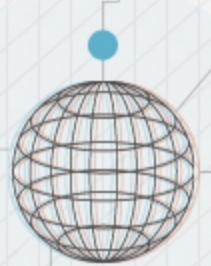


```
$ npm install coffee-script Install them locally
```

```
var coffee = require('coffee-script');
```

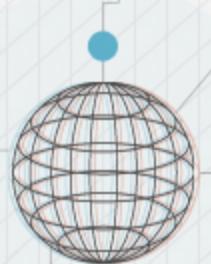


MODULES





# FINDING MODULES



## npm registry

Find packages...

npm registry or browse

Latest Updates	Most Depended
adstream-data grover tint genetics oauth2-node kdtree stateful siq-mesh	7 minutes ago 7 minutes ago 28 minutes ago 44 minutes ago 49 minutes ago 54 minutes ago 1 hour ago 1 hour ago
	underscore coffee-script request express async optimist colors connect

## npm command line

```
$ npm search request
```

## github search

github Search...

Explore      Repositories

Advanced Search

MODULES

## toolbox.no.de

node-toolbox Categories Packages Search

References

- Official Node.js Documentation
- Felix's Node.js Guide
- The Node Beginner Book
- Up and Running with Node.js
- Mastering Node
- Hands-on Node.js

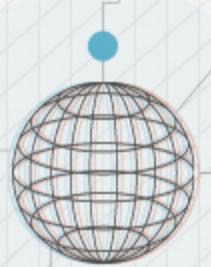
Popular by github

- joyentexpress
- express
- d3
- ender-d3
- blueimp-file-upload-node
- blueimp-file-upload-jquery-ui
- blueimp-file-upload
- coffee-script
- socket.io-wisdom





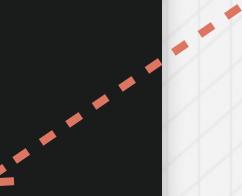
# DEFINING YOUR DEPENDENCIES



my\_app/package.json

```
{  
  "name": "My App",  
  "version": "1",  
  "dependencies": {  
    "connect": "1.8.7"  
  }  
}
```

*version number*



```
$ npm install
```

Installs into the node\_modules directory

my\_app

/ node\_modules /

connect



MODULES





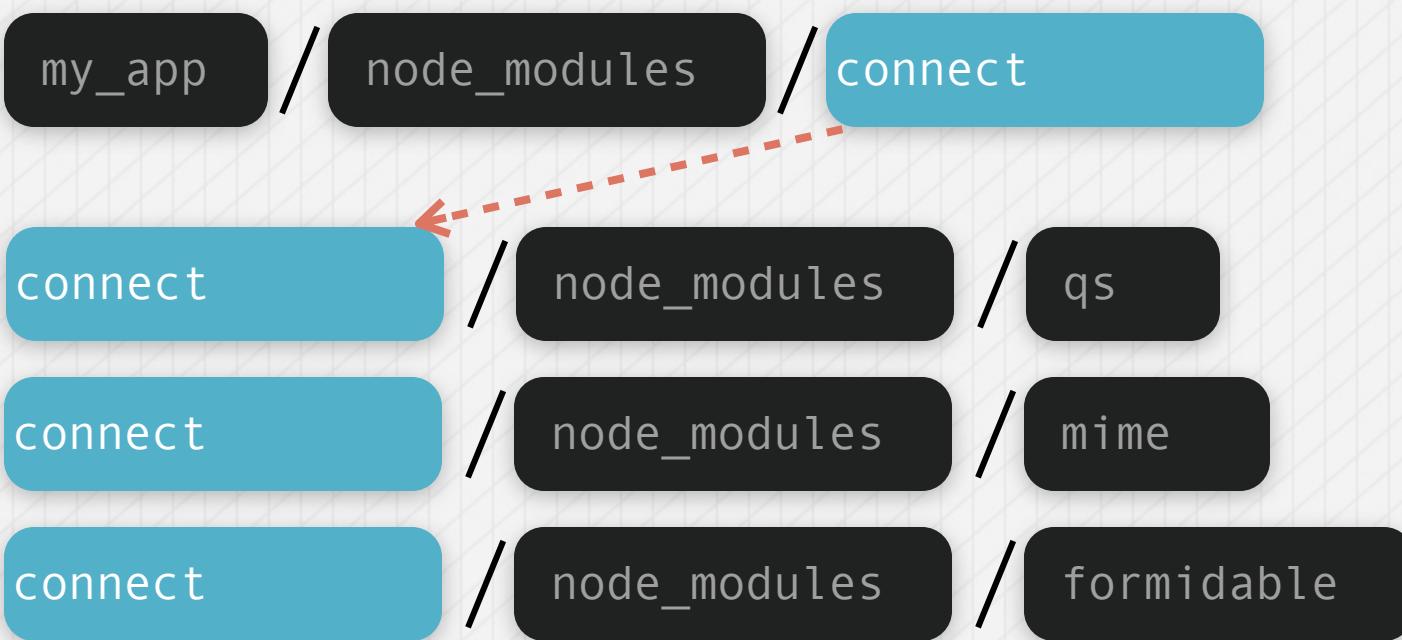
# DEPENDENCIES

my\_app/package.json

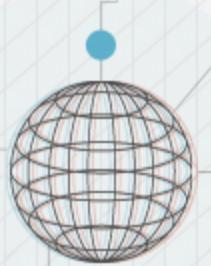
```
"dependencies": {  
  "connect": "1.8.7"  
}
```

No conflicting modules!

Installs sub-dependencies



MODULES





# SEMANTIC VERSIONING

```
"connect": "1.8.7"
```

Major    Minor    Patch

1

8

7

## Ranges

```
"connect": "~1"
```

- ->

$\geq 1.0.0 < 2.0.0$

Dangerous

```
"connect": "~1.8"
```

- ->

$\geq 1.8 < 2.0.0$

API could change

```
"connect": "~1.8.7"
```

- ->

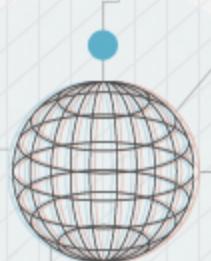
$\geq 1.8.7 < 1.9.0$

Considered safe

<http://semver.org/>

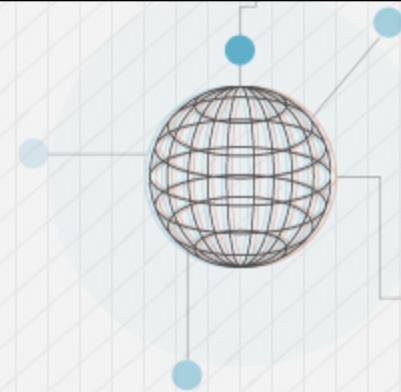


MODULES



# EXPRESS

- LEVEL FIVE -

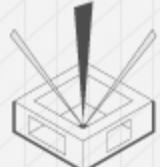




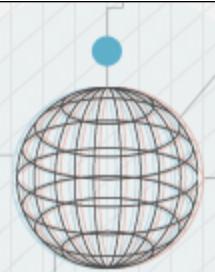
# EXPRESS

“Sinatra inspired web development framework for Node.js -- insanely fast, flexible, and simple”

- Easy route URLs to callbacks
- Middleware (from Connect)
- Environment based configuration
- Redirection helpers
- File Uploads

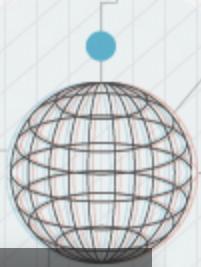


EXPRESS





# INTRODUCING EXPRESS



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

```
$ npm install express
```

```
var app = express.createServer();
```

```
app.get('/', function(request, response) {  
  response.sendfile(__dirname + "/index.html");  
});
```

*root route*

*current directory*

```
app.listen(8080);
```

```
$ curl http://localhost:8080/  
> 200 OK
```

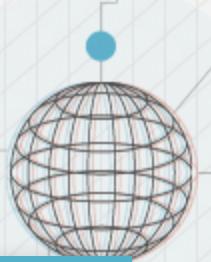


EXPRESS





# EXPRESS ROUTES



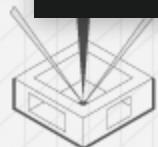
app.js

```
var request = require('request');
var url = require('url');

app.get('/tweets/:username', function(req, response) {
  var username = req.params.username;

  options = {
    protocol: "http:",           get the last 10 tweets for screen_name
    host: 'api.twitter.com',
    pathname: '/1/statuses/user_timeline.json',
    query: { screen_name: username, count: 10}
  }

  var twitterUrl = url.format(options);
  request(twitterUrl).pipe(response);   pipe the request to response
});
```



EXPRESS





# EXPRESS ROUTES



Node.js

level4 — Node.js

Node.js

\$

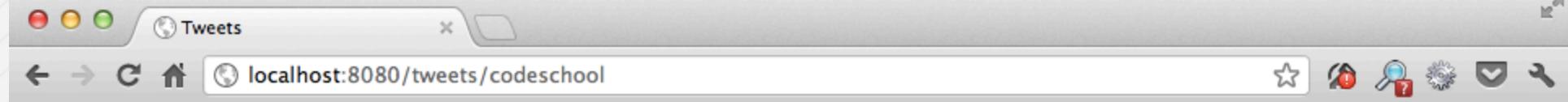


EXPRESS





# EXPRESS + HTML



## Tweets for @codeschool

- @seandevineinc let us know how it goes, and good luck
- @larzconwell Nope, we didn't give those away. The one David has comes from <http://t.co/XrvybxnS> ^OL
- We just released a new Code TV screencast for enrolled members. Part of 1 of @markkendall's jQuery Mobile series. <http://t.co/FstmuYEM>
- ^vc
- We also have stickers..
- Are you at Railsconf? Come by the beginner track room.. we're giving away free Rails for Zombies T-shirts (while they last) #railsconf
- We themed out our Code School store. Check it out <http://t.co/VOZCgorM> ^vc
- Have you gotten your Code School & Zombies t-shirts yet? Check out our \$19 sale this week... <http://t.co/b7JUMfxxy> ^vc

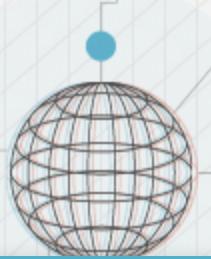


EXPRESS





# EXPRESS TEMPLATES

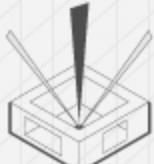


```
app.get('/tweets/:username', function(req, response) {  
  ...  
  request(url, function(err, res, body) {  
    var tweets = JSON.parse(body);  
    response.render('tweets.ejs', {tweets: tweets, name: username});  
  });  
});
```

app.js

```
<h1>Tweets for @<%= name %></h1>  
<ul>  
  <% tweets.forEach(function(tweet){ %>  
    <li><%= tweet.text %></li>  
  <% }); %>  
</ul>
```

tweets.ejs



EXPRESS

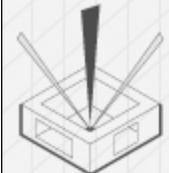
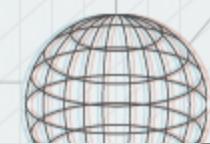




# EXPRESS TEMPLATES

level4 — Node.js

\$



EXPRESS





# TEMPLATE LAYOUTS

```
<h1>Tweets for @<%= name %></h1>
<ul>
  <% tweets.forEach(function(tweet){ %>
    <li><%= tweet.text %></li>
  <% }); %>
</ul>
```

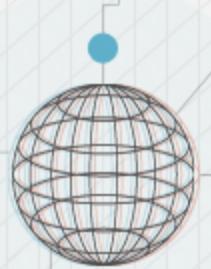
tweets.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>Tweets</title>
  </head>
  <body>
    <% body %>
  </body>
</html>
```

layout.ejs



EXPRESS





# EXPRESS TEMPLATES

level4 — Node.js

\$

I

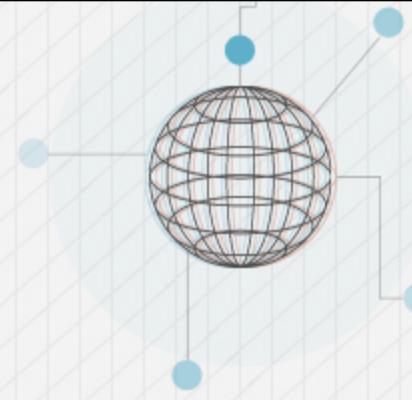


EXPRESS

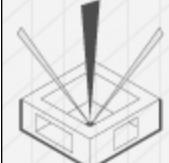




# SOCKET.IO

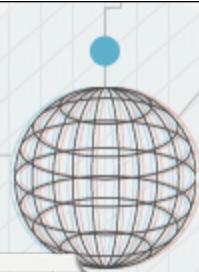


- LEVEL SIX -





# CHATTR



## Hello from Chattr

ERIC

DERRICK

CONNECTED TO CHATTR

*Eric joined the room*

**Derrick**

Hey buddy!

**Eric**

I'm having a great time over here?

*Derrick joined the room*

Type your message

SEND



SOCKET.IO





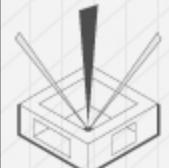
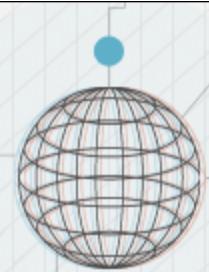
# WEBSOCKETS



browser



traditional server



SOCKET.IO





# WEBSOCKETS



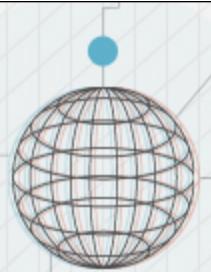
browser

Using duplexed websocket connection

SOCKET.IO

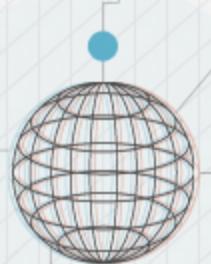


socket.io





# SOCKET.IO FOR WEB SOCKETS



Abstracts websockets with fallbacks

```
$ npm install socket.io
```

```
var socket = require('socket.io');
var app = express.createServer();
var io = socket.listen(app);
```

app.js

```
io.sockets.on('connection', function(client) {
  console.log('Client connected...');
});
```

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var server = io.connect('http://localhost:8080');
</script>
```

index.html

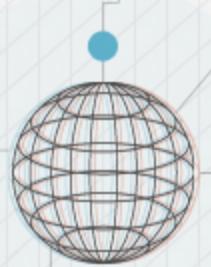


SOCKET.IO





# SENDING MESSAGES TO CLIENT

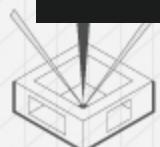


```
io.sockets.on('connection', function(client) {  
  console.log('Client connected...');  
  
  emit the 'messages' event on the client  
  client.emit('messages', { hello: 'world' });  
});
```

app.js

```
<script src="/socket.io/socket.io.js"></script>  
<script>  
  var server = io.connect('http://localhost:8080');  
  server.on('messages', function (data) {  
    alert(data.hello);  
  });  
</script>
```

index.html



SOCKET.IO



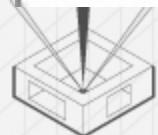


# CHATTR HELLO WORLD

demo — dash

\$

I

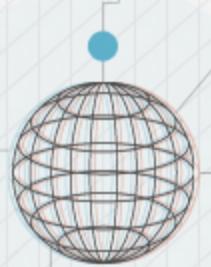


SOCKET.IO





# SENDING MESSAGES TO SERVER



```
io.sockets.on('connection', function(client) {  
  client.on('messages', function (data) {  
    console.log(data);  
  });  
});  
listen for 'messages' events
```

app.js

```
<script>  
  var server = io.connect('http://localhost:8080');  
  $('#chat_form').submit(function(e){  
    var message = $('#chat_input').val();  
    emit the 'messages' event on the server  
    socket.emit('messages', message);  
  });  
</script>
```

index.html



SOCKET.IO





# CHATTR HELLO WORLD

level4 — bash

\$

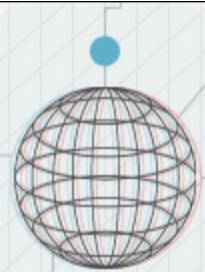


SOCKET.IO





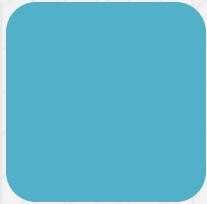
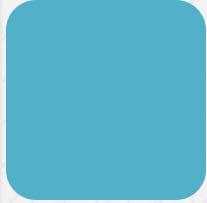
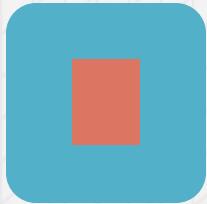
# BROADCASTING MESSAGES



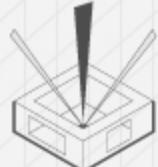
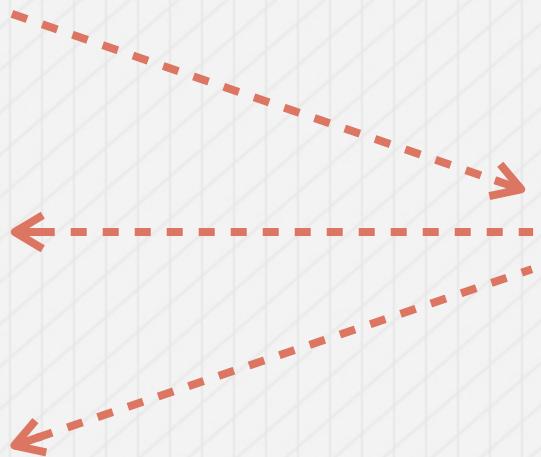
app.js

```
socket.broadcast.emit("message", 'Hello');
```

*clients*



*server*

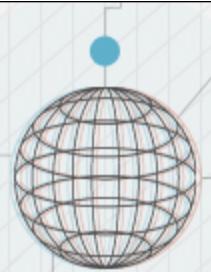


SOCKET.IO





# BROADCASTING MESSAGES



```
io.sockets.on('connection', function(client) {  
  client.on('messages', function (data) {  
    client.broadcast.emit("messages", data);  
  });  broadcast message to all other clients connected  
});
```

app.js

```
<script>
```

```
...
```

```
  server.on('messages', function(data) { insertMessage(data) });  
</script>
```

*insert message into the chat*

index.html



SOCKET.IO

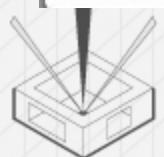




# BROADCASTING MESSAGES

level4 — bash

\$

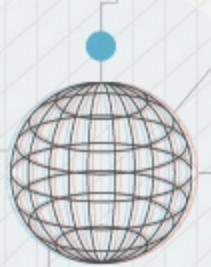


SOCKET.IO





# SAVING DATA ON THE SOCKET



```
io.sockets.on('connection', function(client) {  
  client.on('join', function(name) {  
    client.set('nickname', name);   set the nickname associated  
    with this client  
  });  
});
```

app.js

```
<script>  
  var server = io.connect('http://localhost:8080');  
  server.on('connect', function(data) {  
    $('#status').html('Connected to chattr');  
    nickname = prompt("What is your nickname?");  
  
    server.emit('join', nickname);   notify the server of the  
    users nickname  
  });  
</script>
```

index.html

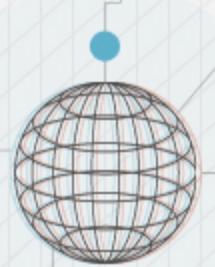


SOCKET.IO





# SAVING DATA ON THE CLIENT



```
io.sockets.on('connection', function(client) {  
  client.on('join', function(name) {  
    client.set('nickname', name);   set the nickname associated  
    with this client  
  });  
  client.on('messages', function(data){  
    get the nickname of this client before broadcasting message  
    client.get('nickname', function(err, name) {  
      client.broadcast.emit("chat", name + ": " + message);  
    });  
    broadcast with the name and message  
  });  
});
```

app.js



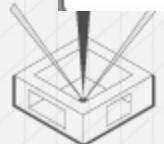
SOCKET.IO





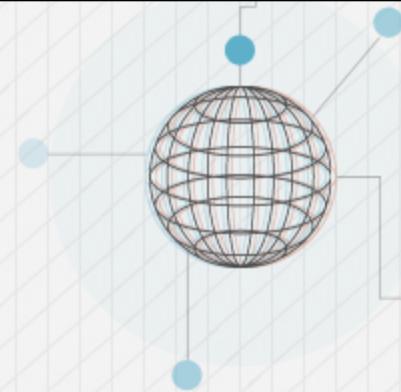
# SAVING DATA ON THE CLIENT

level4 — bash



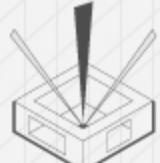
SOCKET.IO





# PERSISTING DATA

- LEVEL SEVEN -





# RECENT MESSAGES

Hello from Chattr

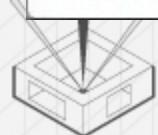
localhost:8080

CONNECTED TO CHATTR

Type your message

SEND

A screenshot of a web browser window titled "Hello from Chattr". The address bar shows "localhost:8080". The main content area displays the text "CONNECTED TO CHATTR". At the bottom, there is a text input field with the placeholder "Type your message" and a "SEND" button. The browser has a dark blue header and a light gray body, with standard Mac OS X window controls.

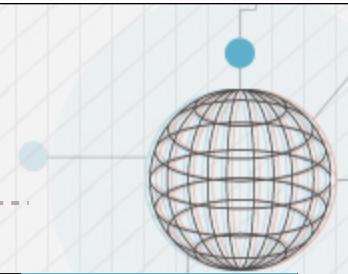


PERSISTING DATA



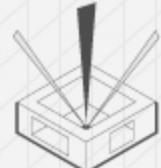


# RECENT MESSAGES



```
io.sockets.on('connection', function(client) {  
  client.on('join', function(name) {  
    client.set('nickname', name);  
    client.broadcast.emit("chat", name + " joined the chat");  
  });  
  client.on("messages", function(message){  
    client.get("nickname", function(error, name) {  
      client.broadcast.emit("messages", name + ": " + message);  
    });  
  });  
});
```

app.js

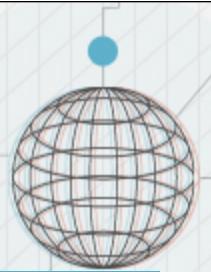


PERSISTING DATA





# STORING MESSAGES



```
var messages = [];
```

*store messages in array*

```
var storeMessage = function(name, data){  
  messages.push({name: name, data: data});  
  if (messages.length > 10) {  
    messages.shift();  
  }  
}
```

*add message to end of array*

*if more than 10 messages long,  
remove the last one*

```
io.sockets.on('connection', function(client) {  
  client.on("messages", function(message){  
    client.get("nickname", function(error, name) {  
      storeMessage(name, message);  
    });  
  });  
});
```

*when client sends a message  
call storeMessage*

app.js

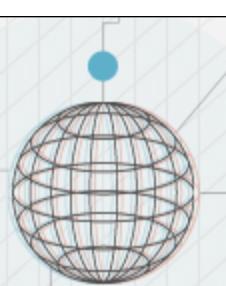


PERSISTING DATA





# EMITTING MESSAGES



app.js

```
io.sockets.on('connection', function(client) {  
  ...  
  client.on('join', function(name) {  
    messages.forEach(function(message) {  
      client.emit("messages", message.name + ": " + message.data);  
    });  iterate through messages array  
  });  and emit a message on the connecting  
});  client for each one
```

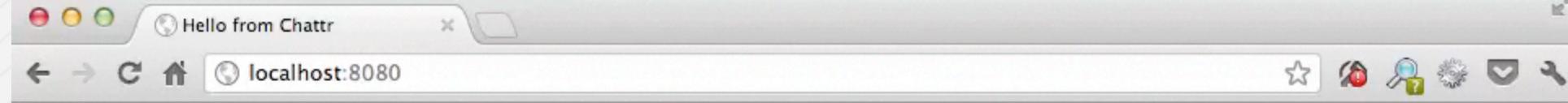


PERSISTING DATA





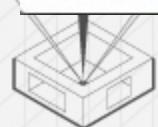
# RECENT MESSAGES



CONNECTED TO CHATTR

Type your message

SEND



PERSISTING DATA





# PERSISTING STORES

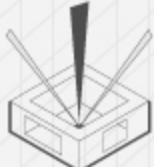
- MongoDB
- CouchDB
- PostgreSQL
- Memcached
- Riak

All non-blocking!

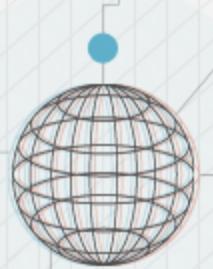


redis

Redis is a key-value store



PERSISTING DATA

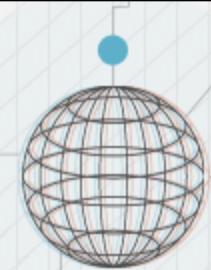




# REDIS DATA STRUCTURES

*data structure*

*commands*



Strings

*SET, GET, APPEND, DECR, INCR...*

Hashes

*HSET, HGET, HDEL, HGETALL...*

Lists

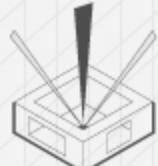
*LPUSH, LREM, LTRIM, RPOP, LINSERT...*

Sets

*SADD, SREM, SMOVE, SMEMBERS...*

Sorted Sets

*ZADD, ZREM, ZSCORE, ZRANK...*

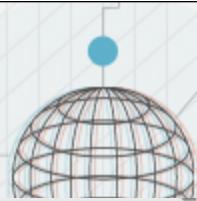


PERSISTING DATA





# REDIS COMMAND DOCUMENTATION



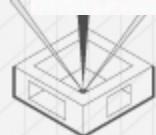
The screenshot shows a browser window with the URL [redis.io](http://redis.io). The page features a dark header with the Redis logo on the left and a navigation menu with links to 'Commands', 'Clients', 'Documentation', 'Community', 'Download', and 'Issues'.

Redis is an open source, advanced **key-value store**. It is often referred to as a **data structure server** since keys can contain strings, hashes, lists, sets and sorted sets.

[Learn more →](#)

## Try it

Ready for a test drive? Check this [interactive tutorial](#) that will walk you through the most important features of Redis.



## PERSISTING DATA

## Download it

[Redis 2.4.13 is the latest stable version](#). Interested in release candidates or unstable versions? [Check the downloads page](#).

What people are saying



Facebook Sets I.P.O.  
Price Range  
<http://t.co/7qTOhWMx>



@tinkertim No more spaces screwing my Redis commands.  
Pretty major to me ;-)



#RedMango #coupon?  
Get a \$2 OFF one  
@coupons.com. Just enter your ZIP code in the upper left! US only.  
<http://t.co/P0i9nvUh>

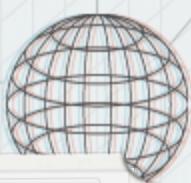


redis (@DIRTYBIITCH)





# NODE REDIS



github

Search...



Explore Gist Blog Help



rubymaverick



mraney / node\_redis

Watch

Fork

1,287

109

Code

Network

Pull Requests

8

Issues

19

Wiki

1

Graphs

redis client for node — [Read more](#)

Clone in Mac

ZIP

HTTP

Git Read-Only

[https://github.com/mranney/node\\_redis.git](https://github.com/mranney/node_redis.git)

Read-Only access

branch: master

Files

Commits

Branches 3

Tags 24

Downloads

Latest commit to the master branch

test.js: Switch to pubsub mode when the number of channels is > 0. [...](#)

 jdavisp3 authored 3 months ago

 commit 874a893c2c 

→ DTrejo committed 4 days ago

[node\\_redis](#) /

name	age	message	history
 examples	6 months ago	Add eval example. [mranney]	
 lib	6 months ago	Better util/sys fallback with try/catch instead of version magic. [mranney]	
 tests	2 months ago	Add failing test. [bnoguchi]	

PERSISTING DATA





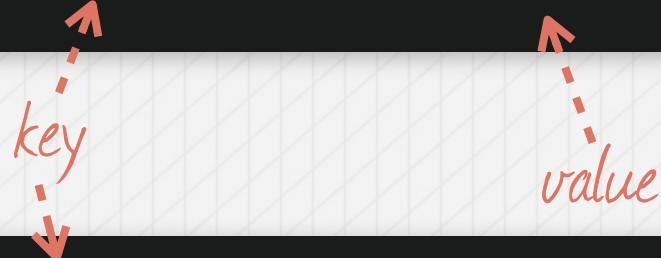
# REDIS

```
$ npm install redis
```

```
var redis = require('redis');
var client = redis.createClient();

client.set("message1", "hello, yes this is dog");
client.set("message2", "hello, no this is spider");
```

key                      value

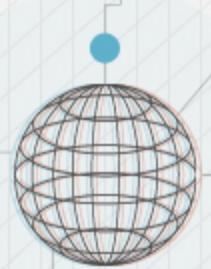


```
client.get("message1", function(err, reply){
  console.log(reply); -----> "hello, yes this is dog"
});
```

## commands are non-blocking

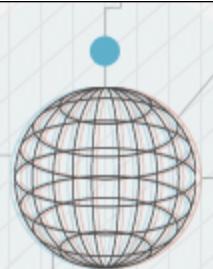


PERSISTING DATA





# REDIS LISTS: PUSHING



## Add a string to the “messages” list

```
var message = "Hello, this is dog";
client.lpush("messages", message, function(err, reply){
  console.log(reply); - - -> "1" replies with list length
});
```

## Add another string to “messages”

```
var message = "Hello, no this is spider";
client.lpush("messages", message, function(err, reply){
  console.log(reply); - - -> "2"
});
```



PERSISTING DATA





# REDIS LISTS: RETRIEVING

## Using LPUSH & LTRIM

```
var message = "Oh sorry, wrong number";
client.lpush("messages", message, function(err, reply){
  client.ltrim("messages", 0, 1);    trim keeps first two strings
});                                and removes the rest
```

## Retrieving from list

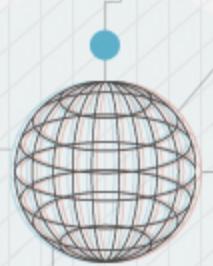
```
client.lrange("messages", 0, -1, function(err, messages){
  console.log(messages);    replies with all strings in list
})
```



```
["Hello, no this is spider", "Oh sorry, wrong number"]
```

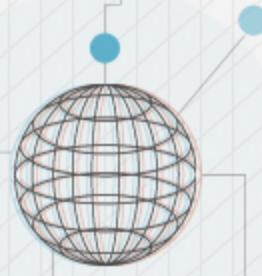


PERSISTING DATA





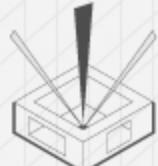
# CONVERTING MESSAGES TO REDIS



```
var storeMessage = function(name, data){  
  messages.push({name: name, data: data});  
  
  if (messages.length > 10) {  
    messages.shift();  
  }  
}
```

app.js

Let's use the List data-structure



PERSISTING DATA





# CONVERTING STOREMESSAGE



app.js

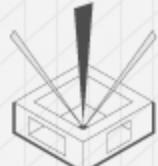
```
var redisClient = redis.createClient();

var storeMessage = function(name, data){
  var message = JSON.stringify({name: name, data: data});
    need to turn object into string to store in redis

  redisClient.lpush("messages", message, function(err, response) {
    redisClient.ltrim("messages", 0, 10);
  });
}
```



keeps newest 10 items

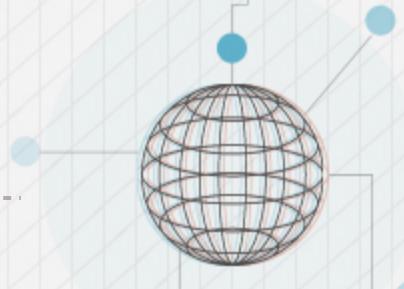


PERSISTING DATA



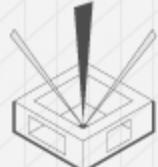


# OUTPUT FROM LIST



```
client.on('join', function(name) {  
  messages.forEach(function(message) {  
    client.emit("messages", message.name + ": " + message.data);  
  });  
});
```

app.js



PERSISTING DATA





# OUTPUT FROM LIST



app.js

```
client.on('join', function(name) {  
  redisClient.lrange("messages", 0, -1, function(err, messages){  
    messages = messages.reverse();    reverse so they are emitted  
    in correct order  
    messages.forEach(function(message) {  
      message = JSON.parse(message); parse into JSON object  
      client.emit("messages", message.name + ": " + message.data);  
    });  
  });  
});  
});
```



PERSISTING DATA



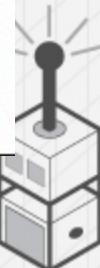


# IN ACTION

A screenshot of a web browser window titled "Hello from Chattr". The address bar shows "localhost:8080". The page content area displays the text "CONNECTED TO CHATTR". At the bottom, there is a text input field with the placeholder "Type your message" and a dark grey "SEND" button to its right.

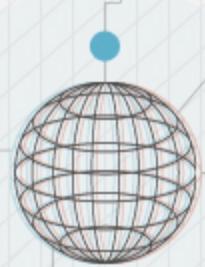


PERSISTING DATA





# CURRENT CHATTER LIST



## Sets are lists of unique data

add & remove members of the names set

DOG  
SPIDER  
GREGG

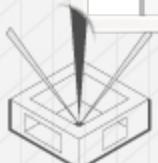
```
client.sadd("names", "Dog");
client.sadd("names", "Spider");
client.sadd("names", "Gregg");
```

```
client.srem("names", "Spider");
```

reply with all members of set

```
client.smembers("names", function(err, names){
  console.log(names);
});
```

["Dog", "Gregg"]



PERSISTING DATA





# ADDING CHATTERS

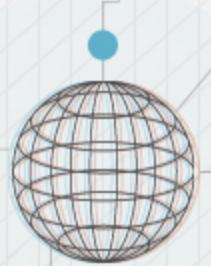
```
client.on('join', function(name){  
  
    notify other clients a chatter has joined  
    client.broadcast.emit("add chatter", name);  
  
    redisClient.sadd("chatters", name);  
});  
    add name to chatters set
```

app.js

```
server.on('add chatter', insertChatter);  
  
var insertChatter = function(name) {  
    var chatter = $('<li>' + name + '</li>').data('name', name);  
    $('#chatters').append(chatter);  
}
```

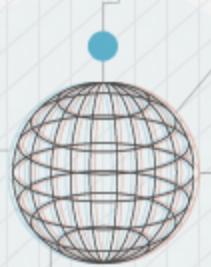
index.html

PERSISTING DATA





# ADDING CHATTERS (CONT)



```
client.on('join', function(name){  
  notify other clients a chatter has joined  
  client.broadcast.emit("add chatter", name);  
  
  redisClient.smembers('names', function(err, names) {  
    names.forEach(function(name){  
      client.emit('add chatter', name);  
    });  
  });  
});  
  
redisClient.sadd("chatters", name);  
});  
add name to chatters set
```

**app.js**

*emit all the currently logged in chatters to the newly connected client*



PERSISTING DATA





# REMOVING CHATTERS

*remove chatter when they disconnect from server*

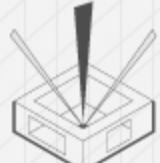
```
client.on('disconnect', function(name){  
  client.get('nickname', function(err, name){  
    client.broadcast.emit("remove chatter", name);  
  
    redisClient.srem("chatters", name);  
  });  
});
```

app.js

```
server.on('remove chatter', removeChatter);
```

```
var removeChatter = function(name) {  
  $('#chatters li[data-name=' + name + ']').remove();  
}
```

index.html



PERSISTING DATA

