



# MongoDB

MapReduce





- But first, some functional programming
- Using Ruby blocks/closures
- Will help understand how MapReduce works









Functional programming for PHP programmers.





Functional programming for PHP programmers.

Javascripters, Pythonistas and Rubyists feel free to doze off for a bit :P



- What is MapReduce?
- A method of aggregation that can easily be parallelized across multiple servers
  - It splits up a problem
  - Sends chunks of it to different machines
  - Lets each machine solve its part of the problem
- When all machines are finished, they merge all the pieces of the solution back into a full solution



- Caveats:
- Can be slow
- Not meant to be used "real-time"
- Usually run as a background job, which creates a collection of results, which you then query
- Think batch processing



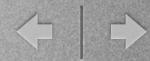
Example: count number of tags db.things.insert( { \_\_id : I, tags : ['dog', db.things.insert( { \_\_id : 2, tags : ['cat'] } ); db.things.insert( { \_\_id : 3, tags : ['mouse', 'cat', db.things.insert( { \_id : 4, tags : [] } );



```
> mr = db.things.mapReduce(m, r, { out : "myoutput" } );
  "result": "myoutput",
  "timeMillis": 12,
  "counts":{
       "input" : 4,
       "emit": 6,
       "output":3
  "ok": I,
```



- "result": "myoutput",
- The name of the collection the MapReduce results were stored in
- "timeMillis": 12
- How long the operation took, in milliseconds



- "counts": { ... }
- This embedded document contains three keys:
  - "input": 6
    The number of documents sent to the map function.
  - "emit": 14
    The number of times emit was called in the map function.
  - "output" : 5
  - The number of documents created in the result collection. "counts" is mostly useful for debugging.