MICROSERVICES & API'S (& NODE.JS)

Al Graham (@blogtackular)

</PLUG>

WHAT THIS IS

WHAT THIS ISN'T

WHAT ARE MICROSERVICES?

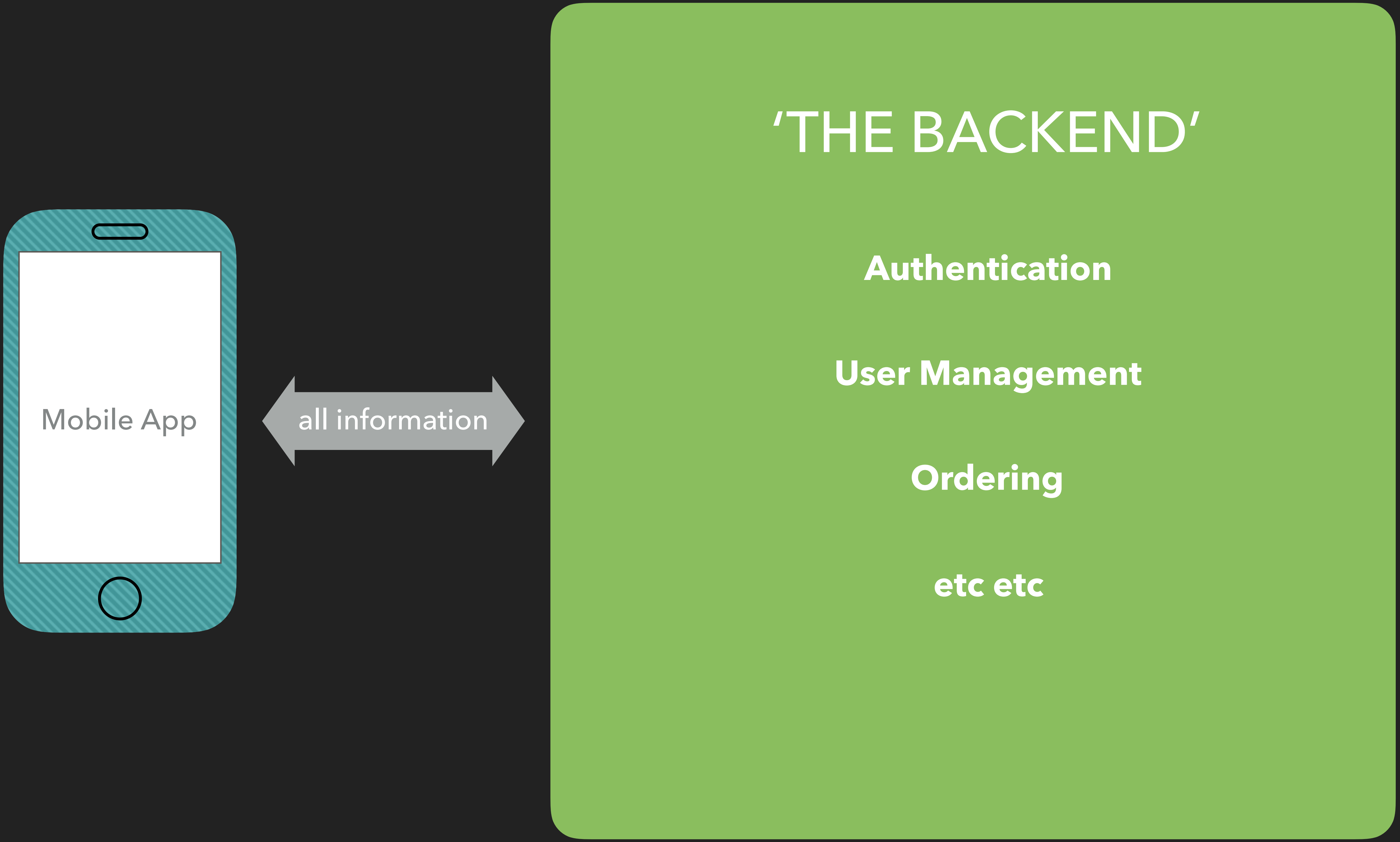
COMPLEX SERVICE-ASSEMBLIES ARE ABSTRACTED BEHIND (A) SIMPLE URI INTERFACE. ANY SERVICE, AT ANY GRANULARITY, CAN BE EXPOSED

Dr Peter Rodgers

CloudComputing Expo, 2005

HUH?

THE OLD MONOLITHS



THE 'OLD' WAY

MONOLITHIC SERVICES, ALL TIGHTLY COUPLED

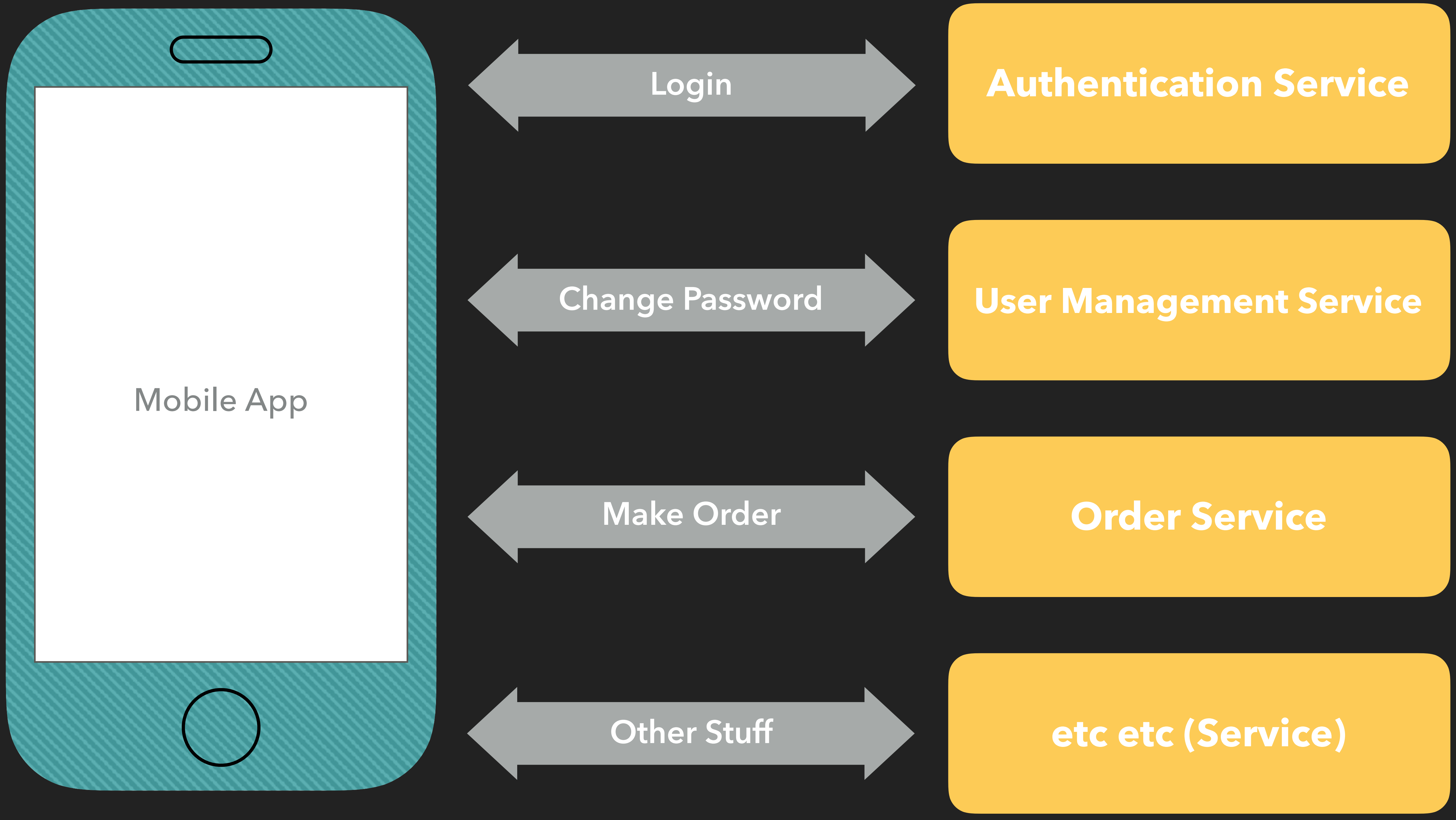
HARD TO SCALE

HARD TO CHANGE

HARD TO DEPLOY

… JUST **HARD**

MICROSERVICES



WE HAVE LOOSELY COUPLED, DISCRETE SERVICES

EASY TO SCALE –SPIN UP MORE INSTANCES!

EASY TO CHANGE, AS NO IMPACT ON THE OTHERS

EASY TO DEPLOY

… LOTS **EASIER!**

BUT!

THERE ARE PROBLEMS…

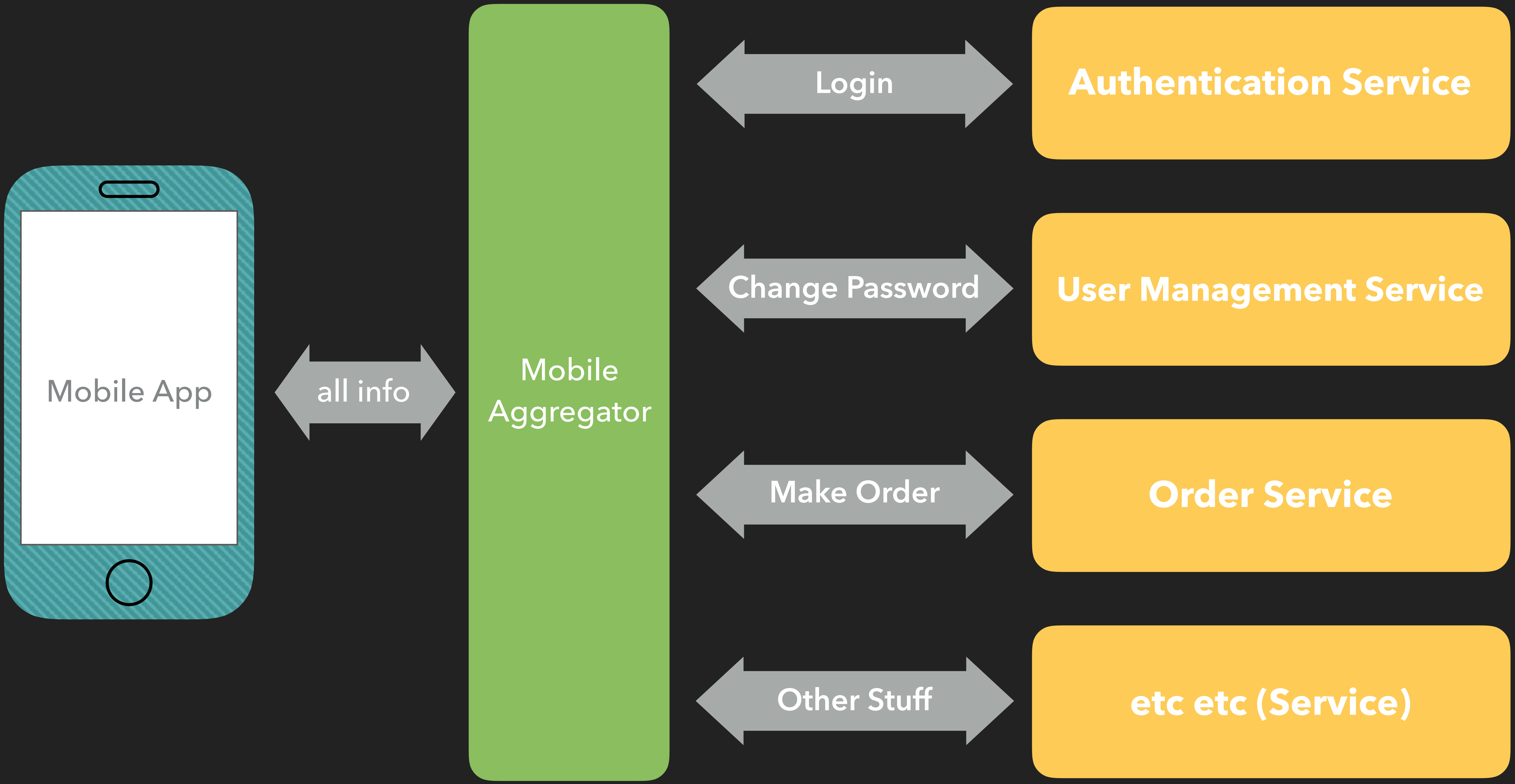
DISCRETE SERVICES NEED GOOD, STABLE API'S

LOTS OF ENDPOINTS (AND URL'S) TO MANAGE

NEED SOME WAY TO MAINTAIN STATE

… BUT THERE ARE **SOLUTIONS**

MICROSERVICES - AGGREGATE



THE MICROSERVICE WAY

SOLVES THE PROBLEMS…

~~DISCRETE SERVICES NEED GOOD, STABLE API'S~~

AGGREGATOR CAN HANDLE ANY CHANGES

~~LOTS OF ENDPOINTS (AND URL'S) TO MANAGE~~

EVERYTHING GOES THROUGH THE AGGREGATOR

~~NEED SOME WAY TO MAINTAIN STATE~~

JASON WEB TOKENS (JWT) ARE YOUR BEST FRIEND

… SO HOW CAN I **DO THIS?**

NODE.JS

NODE.JS HELPS US

QUICKLY CREATE SERVICES

MODULARISE SAID SERVICES TO DECOUPLE THEM

QUICKLY DEPLOY SERVICES

KEEP EVERYTHING FAMILIAR AND 'JS-Y'

…SHOW ME

CREATE A WEBSERVICE

```
$ npm install express
```

Setup the scaffolding…

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

app.listen(8001, '0.0.0.0', function() {
    console.log("App started at: " + new Date() + " on port: 8001");
});
```

server.js

Add some endpoints…

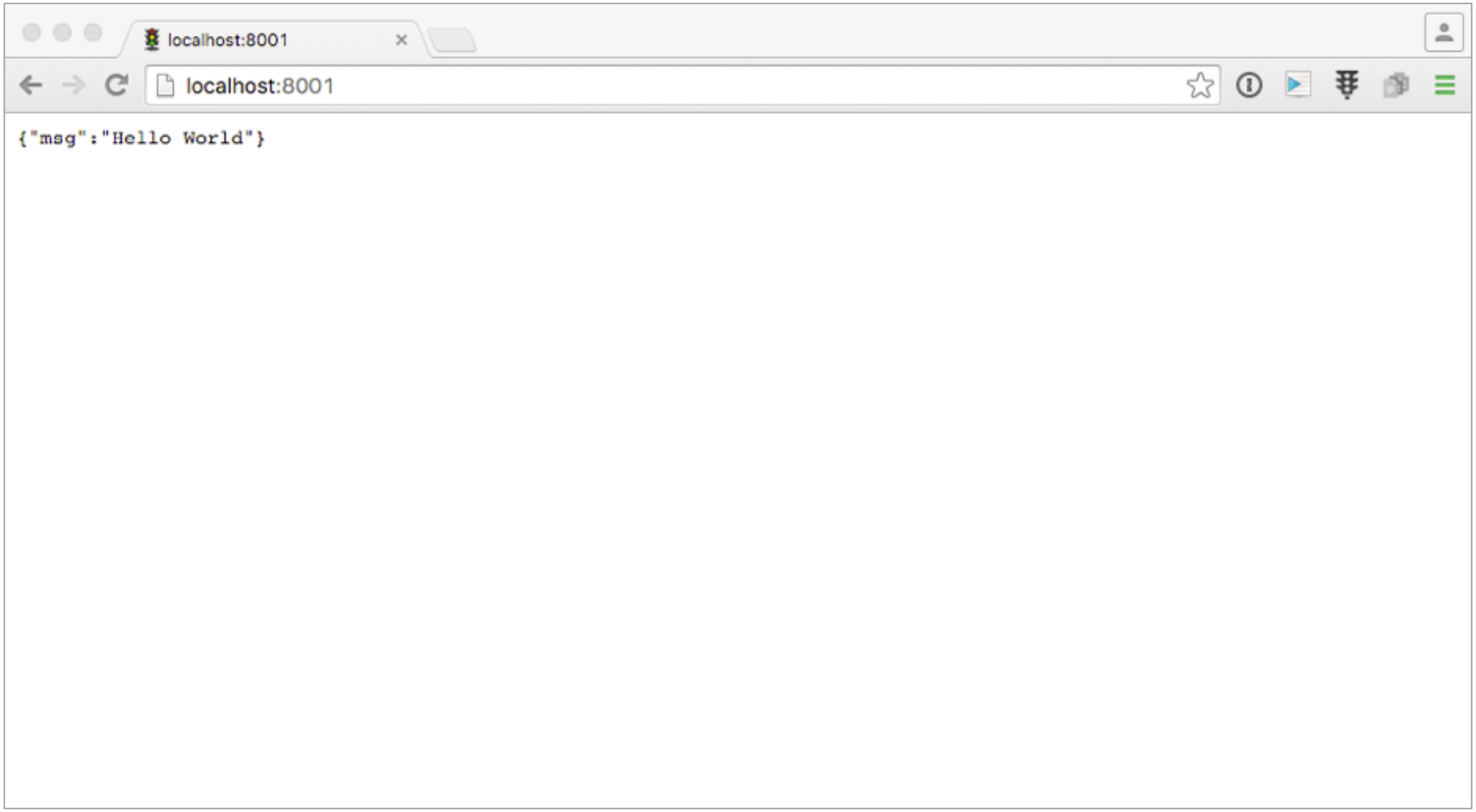
```
var express = require('express'),
    app = express();

app.get('/', function(req, res) {
  console.log(new Date(), 'In hello route GET / req.query=', req.query);
  var world = req.query && req.query.hello ? req.query.hello : 'World';
  res.json({
    msg: 'Hello ' + world
  });
});

app.post('/', function(req, res) {
  console.log(new Date(), 'In hello route POST / req.body=', req.body);
  var world = req.body && req.body.hello ? req.body.hello : 'World';
  res.json({
    msg: 'Hello ' + world
  });
});

app.listen(8001, '0.0.0.0', function() {
    console.log("App started at: " + new Date() + " on port: 8001");
});
```

server.js

CREATE A WEBSERVICE

Run it…

```
$ node server.js
```



```
{"msg":"Hello World"}
```

MONOLITH!

Create hello.js module

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

function helloRoute() {
    var hello = new express.Router();

    hello.get('/', function(req, res) {
        console.log(new Date(), 'In hello route GET / req.query=', req.query);
        var world = req.query && req.query.hello ? req.query.hello : 'World';
        res.json({
            msg: 'Hello ' + world
        });
    });

    hello.post('/', function(req, res) {
        console.log(new Date(), 'In hello route POST / req.body=', req.body);
        var world = req.body && req.body.hello ? req.body.hello : 'World';
        res.json({
            msg: 'Hello ' + world
        });
    });

    return hello;
}

module.exports = helloRoute;
```

hello.js

ADD THE MODULE

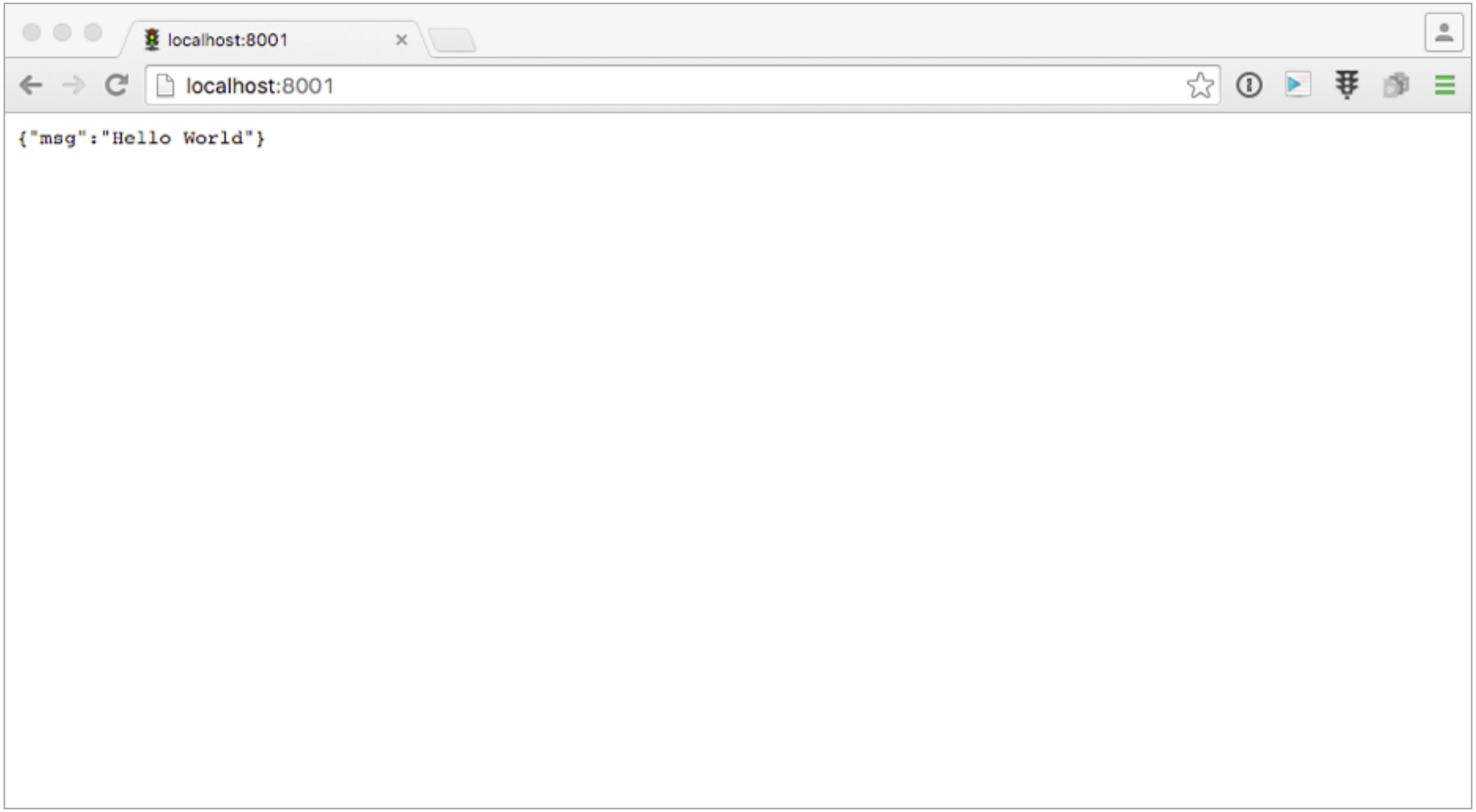
include the module…

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

app.use('/', require('./hello.js')());

app.listen(8001, '0.0.0.0', function() {
    console.log("App started at: " + new Date() + " on port: 8001");
});
```

server.js

Run it again!

```
$ node server.js
```



```
{"msg":"Hello World"}
```

THIS IS A SIMPLE EXAMPLE

EXPRESS ALLOWS YOU TO EASILY CREATE POST, DELETE, GET, ETC

ADD SOCKETS, EASY!

DEPLOY STEPS - SIMPLE WITH NPM & PACKAGE.JSON

MOVE TO DIFFERENT SERVERS, THE REQUEST MODULE SORTS YOU OUT!

API’S

WHICH VERB SHOULD I USE

WHICH RESPONSE SHOULD I SEND

WHAT TO RETURN ON ERRORS

HOW DO I DOCUMENT

WHICH VERBS?

Lots of choice

Is a ‘logout’ a POST, or a DELETE?
You’re deleting a session, but not a user…

Should fetching data *always* be a GET?
What if you know you’ll always need params -url params are unwieldy

WHICH RESPONSE?

If params are missing is it a:

422 Unprocessable Entity
The request was well-formed but was unable to be followed due to semantic errors.

or

400 Bad Request
The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

or

412 Precondition Failed
The server does not meet one of the preconditions that the requester put on the request.

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

both a blessing, and a curse!

THERE'S ONLY ONE THING FOR IT

DOCUMENT

DOCUMENT

DOCUMENT

DOCUMENT

NOT HARD

NODE TO THE RESCUE AGAIN

Lots of modules/libs out there

DOCCO (HTTPS://JASHKENAS.GITHUB.IO/DOCCO/)

NODE-DOC (HTTPS://WWW.NPMJS.COM/PACKAGE/NODE-DOC)

SWAGGER (HTTPS://GITHUB.COM/SWAGGER-API/SWAGGER-NODE)

API-BLUEPRINT (HTTPS://APIBLUEPRINT.ORG/)

WRAP UP

MICROSERVICES ARE GREAT

THEY'RE REALLY JUST FANCY MODULES

MODULES ARE *REALLY* GREAT

THIS IS THE TIP OF THE ICEBERG

THANKS!

Al Graham, @blogtackular