Everything You *Ever* Wanted to Know About Authentication in Node.js

@rdegges

I’m Randall Degges

Developer ~~Evangelist~~ at Stormpath

Python / Node / Go Hacker

A Simple Caller ID API

OpenCNAM provides a simple, elegant, and RESTful API to get Caller ID data. Our service is built for programmers like you, who want simple access to Caller ID information.

Our API can be integrated into any application in a matter of minutes. Getting started requires no registration or signup. All you need to do is query our API endpoint!

You may want to read our docs, check out our plans, or **give us a try right now**:

```
$ curl https://api.opencnam.com/v2/phone/+16502530000
GOOGLE INC
```

GET STARTED NOW

- Build a simple Node.js site.
- Store user accounts in MongoDB.
- Register and login users.
- Safely store user passwords using bcrypt.
- Enforce authentication rules on pages.
- HTTP authentication.

https://github.com/rdegges/svcc-auth

https://speakerdeck.com/rdegges

0x00 - Getting Set Up


node
JS
™

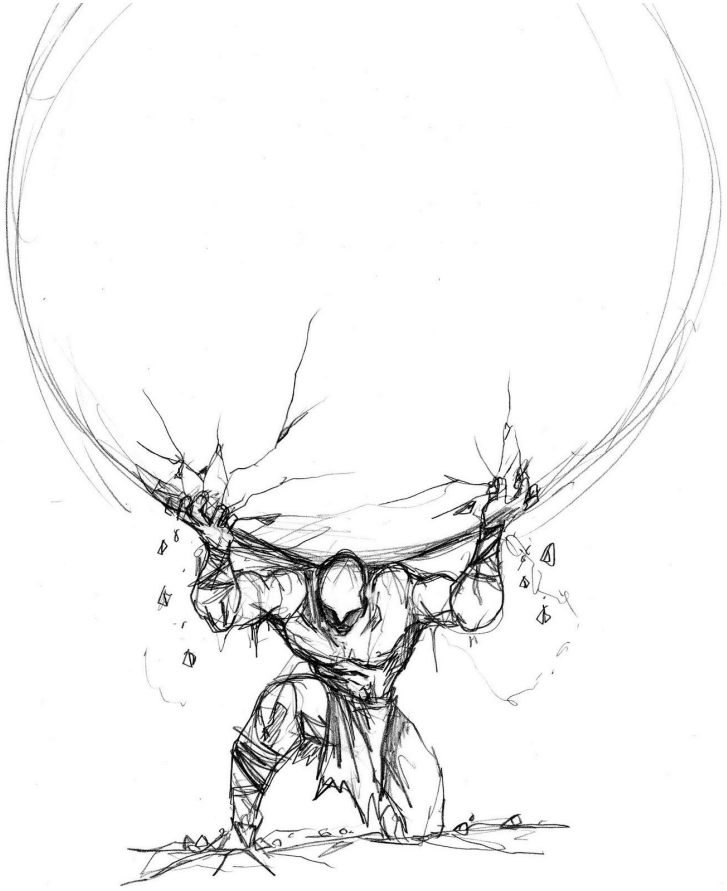

mongoDB

Prep the App

```
$ mkdir views
$ touch app.js
$ touch views/base.jade
$ touch views/index.jade
$ touch views/register.jade
$ touch views/login.jade
$ touch views/dashboard.jade
```

Install Dependencies


```
$ npm install express
$ npm install jade
```



Base Templates

base.jade

```
block vars
doctype html
html
  head
    title SVCC Auth | #{title}
  body
    block body
```



```
<!DOCTYPE html>
<html>
  <head>
    <title>SVCC Auth | </title>
  </head>
  <body>
  </body>
</html>
```

index.jade

```
extends base

block vars
  - var title = 'Home'

block body
  h1 SVCC Auth!
  p.
    Welcome to the SVCC Auth! home page.  Please
    <a href="/register">register</a> or <a href="/login">login</a> to continue!
```

register.jade

```
extends base

block vars
  - var title = 'Register'

block body
  h1 Create an Account
  form(method="post")
    span First Name:
    input(type="text", name="firstName", required=true)
    br
    span Last Name:
    input(type="text", name="lastName", required=true)
    br

    span Email:
    input(type="email", name="email", required=true)
    br

    span Password:
    input(type="password", name="password", required=true)
    br

    input(type="submit")
```

login.jade

```
extends base

block vars
  - var title = 'Login'

block body
  h1 Log Into Your Account

  if error
    p ERROR: #{error}
```

```
form(method="post")
  span Email:
  input(type="email", name="email", required=true)
  br

  span Password:
  input(type="password", name="password", required=true)
  br

  input(type="submit")
```

dashboard.jade

```
extends base

block vars
  - var title = 'Dashboard'

block body
  h1 Dashboard
  p.
    Welcome to your dashboard! You are now logged in.
```

Base App

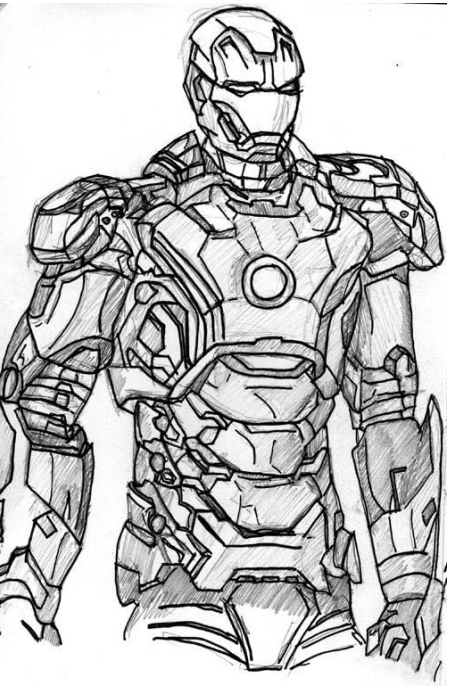
app.js

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

var app = express();

app.set('view engine', 'jade');

app.get('/', function(req, res) {
  res.render('index.jade');
});
```

```
app.get('/register', function(req, res) {
  res.render('register.jade');
});

app.get('/login', function(req, res) {
  res.render('login.jade');
});

app.get('/dashboard', function(req, res) {
  res.render('dashboard.jade');
});

app.listen(3000);
```

Now… Run it!

```
$ node app.js
```



0x01 - HTML

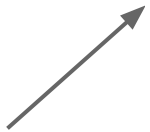
Forms!

```
<form method="post">
   First Name: <input type="text" name="firstName" required/>
   Last Name: <input type="text" name="lastName" required/>
   Email: <input type="email" name="email" required/>
   Password: <input type="password" name="password" required/>
  <input type="submit"/>
</form>
```

Form Data

```
$ npm install body-parser
```

```
// app.js
var bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended: true }));

app.post('/register', function(req, res) {
  res.json(req.body);
});
```



```
{
    firstName: "Randall",
    lastName: "Degges",
    email: "r@rdegges.com",
    password: "woot!"
}
```

0x02 - Databases

MongoDB!

```
$ sudo mongod &
$ mongo
MongoDB shell version: 2.6.2
connecting to: test
Server has startup warnings:
2014-10-11T17:12:22.963-0700 [initandlisten]
2014-10-11T17:12:22.963-0700 [initandlisten]
** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
>
```

Basics

```
> use testdb;
switched to db testdb
> show collections;
> db.users.insert({ email: 'r@rdegges.com', password: 'woot' });
WriteResult({ "nInserted" : 1 })
> db.users.find();
{ "_id" : ObjectId("543a2c005fe787e049f1e3ea"), "email" : "r@rdegges.com", "password" : "woot" }
>
```

mongoose (ORM)

```
$ npm install mongoose
```

```
// app.js
var mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/svcc');
```

mongoose Models

```
var Schema = mongoose.Schema;
var ObjectId = Schema.ObjectId;

var User = mongoose.model('User', new Schema({
  id:           ObjectId,
  firstName:    String,
  lastName:     String,
  email:        { type: String, unique: true },
  password:     String,
}));
```

Creating Users

```
app.post('/register', function(req, res) {
  var user = new User({
    firstName:  req.body.firstName,
    lastName:   req.body.lastName,
    email:      req.body.email,
    password:   req.body.password,
  });
  user.save(function(err) {
    if (err) {
      var error = 'Something bad happened! Please try again.';

      if (err.code === 11000) {
        error = 'That email is already taken, please try another.';
      }

      res.render('register.jade', { error: error });
    } else {
      res.redirect('/dashboard');
    }
  });
});
```

Verifying

```
> db.users.find();
{ "_id" : ObjectId("543a2f00e20ba7d946688eab"), "firstName"
: "Randall", "lastName" : "Degges", "email" : "r@rdegges.
com", "password" : "woot!", "__v" : 0 }
>
```

Logging in Users

```
app.post('/login', function(req, res) {
  User.findOne({ email: req.body.email }, function(err, user) {
    if (!user) {
      res.render('login.jade', { error: "Incorrect email / password." });
    } else {
      if (req.body.password === user.password) {
        res.redirect('/dashboard');
      } else {
        res.render('login.jade', { error: "Incorrect email / password."  });
      }
    }
  });
});
```

Recap!

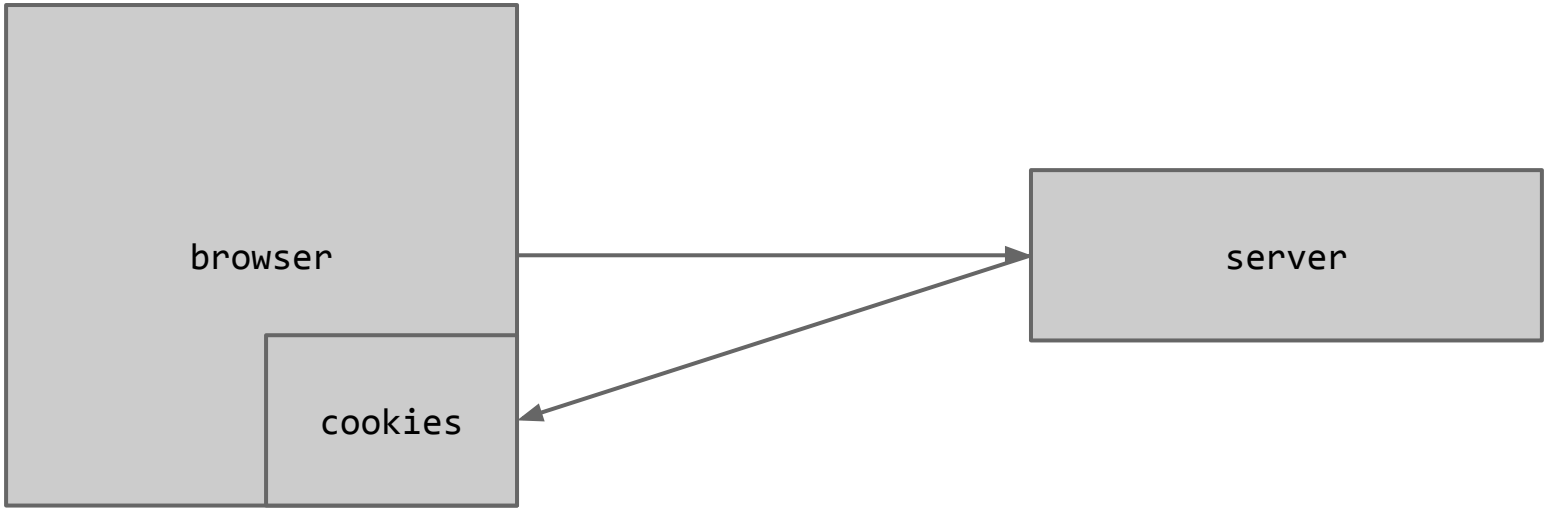
0x03 - Sessions

The Idea

- firstName
- lastName
- email
- etc.
- *(not password)*

- retrieve identity from session
- verify / update
- process request

Cookies!



Reading Cookies

Creating Cookies

```
Set-Cookie: session=r@rdegges.com
```

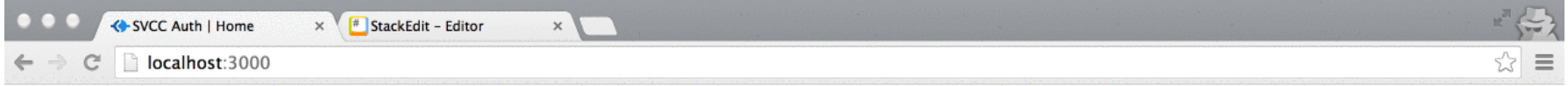
client-sessions

```
$ npm install client-sessions
```

```
var session = require('client-sessions');

app.use(session({
  cookieName: 'session',
  secret: 'some_random_string',
  duration: 30 * 60 * 1000,
  activeDuration: 5 * 60 * 1000,  // optional
}));
```

Using Sessions

```
app.post('/login', function(req, res) {
  User.findOne({ email: req.body.email }, function(err, user) {
    if (!user) {
      res.render('login.jade', { error: "Incorrect email /
password." });
    } else {
      if (req.body.password === user.password) {
        req.session.user = user.email;
        res.redirect('/dashboard');
      } else {
        res.render('login.jade', { error: "Incorrect email /
password."  });
      }
    }
  });
});
```



SVCC Auth!

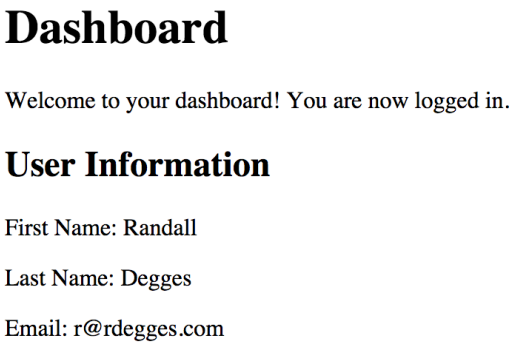
Welcome to the SVCC Auth! home page. Please register or login to continue!

Improving /dashboard

```
app.get('/dashboard', function(req, res) {
  if (req.session && req.session.user) {
    User.findOne({ email: req.session.user }, function(err, user) {
      if (!user) {
        req.session.reset();
        res.redirect('/login');
      } else {
        res.locals.user = user;
        res.render('dashboard.jade');
      }
    });
  } else {
    res.redirect('/login');
  }
});
```

Using Session Info

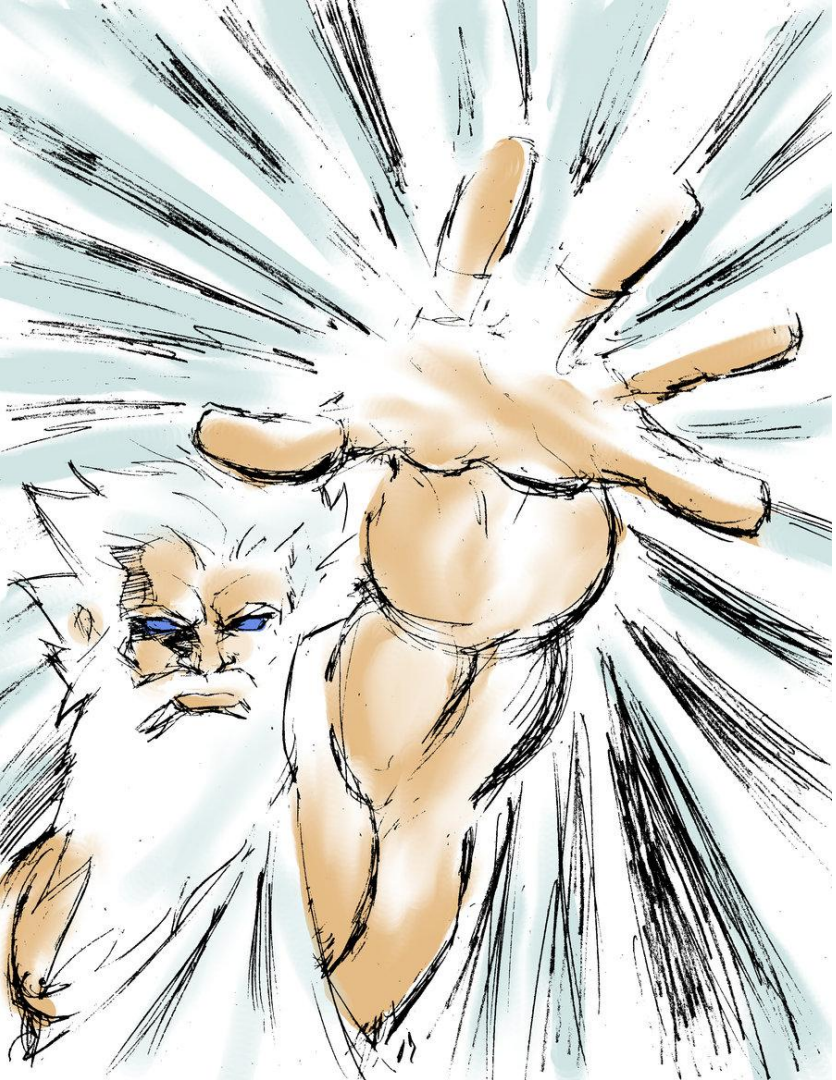
```
extends base

block vars
  - var title = 'Dashboard'

block body
  h1 Dashboard
  p.
    Welcome to your dashboard! You are now logged in.

  h2 User Information
  p First Name: #{user.firstName}
  p Last Name:  #{user.lastName}
  p Email:      #{user.email}
```



0x04 - Storing Passwords

Current User Data

```
{
  "_id" : ObjectId("543a2f00e20ba7d946688eab"),
  "firstName" : "Randall",
  "lastName" : "Degges",
  "email" : "r@rdegges.com",
  "password" : "woot!",
  "__v" : 0
}
```



Hashing!

- md5
- sha256
- **bcrypt**
- scrypt
- etc.

bcrypt (pseudo)

```
var password = 'hi';
var hash = bcrypt(password);

console.log(hash);
// $2a$10$uS.pE0aS0NlsgbvLd6EruO5VDKllinIZLF3C84OYzWHFiyKYfZVXy
```

Improving /register

```
$ npm install bcryptjs
```

```
var bcrypt = require('bcryptjs');

app.post('/register', function(req, res) {
  var salt = bcrypt.genSaltSync(10);
  var hash = bcrypt.hashSync(req.body.password, salt);

  var user = new User({
    firstName:  req.body.firstName,
    lastName:   req.body.lastName,
    email:      req.body.email,
    password:   hash,
  });
```

```
user.save(function(err) {
    if (err) {
      var error = 'Something bad happened! Please try
again.';

      if (err.code === 11000) {
        error = 'That email is already taken, please try
another.';
      }

      res.render('register.jade', { error: error });
    } else {
      req.session.user = user.email;
      res.redirect('/dashboard');
    }
  });
});
```

Improving /login

```
app.post('/login', function(req, res) {
  User.findOne({ email: req.body.email }, function(err, user) {
    if (!user) {
      res.render('login.jade', { error: "Incorrect email / password." });
    } else {
      if (bcrypt.compareSync(req.body.password, user.password)) {
        req.session.user = user.email;
        res.redirect('/dashboard');
      } else {
        res.render('login.jade', { error: "Incorrect email / password."  });
      }
    }
  });
});
```

Improved User

```
{
  "_id" : ObjectId("543a991f8fea0e494e4c0bb1"),
  "firstName" : "Randall",
  "lastName" : "Degges",
  "email" : "r@rdegges.com",
  "password" : "$2a$10$uS.pE0aS0NlsgbvLd6EruO5VDKllinIZLF3C84OYzWHFiyKYfZVXy",
  "__v" : 0
}
```

0x05 - Middleware

Smart User Middleware

```
app.use(function(req, res, next) {
  if (req.session && req.session.user) {
    models.User.findOne({ email: req.session.user }, function(err, user) {
      // if a user was found, make the user available
      if (user) {
        req.user = user;
        req.session.user = user.email;   // update the session info
        res.locals.user = user;          // make the user available to templates
      }
      next();
    });
  } else {
    next();  // if no session is available, do nothing
  }
});
```

Force Authentication

```
function requireLogin(req, res, next) {
  // if this user isn't logged in, redirect them to
  // the login page
  if (!req.user) {
    res.redirect('/login');
  // if the user is logged in, let them pass!
  } else {
    next();
  }
};

app.get('/dashboard', requireLogin, function(req, res) {
  // ...
});
```

0x06 - CSRF

Let’s Say...

```
<!-- bank.com/withdraw -->
<form>
  <input type="text" name="account"/>
  <input type="text" name="amount"/>
  <input type="text" name="for"/>
</form>
```

(cross site request forgery)

Hey Randall,

Check out this picture of my dog!

```
<img src="http://bank.com/withdraw?
account=Randall&amp;amount=1000000&amp;
for=BadGuy">
```

:(

CSRF Protection

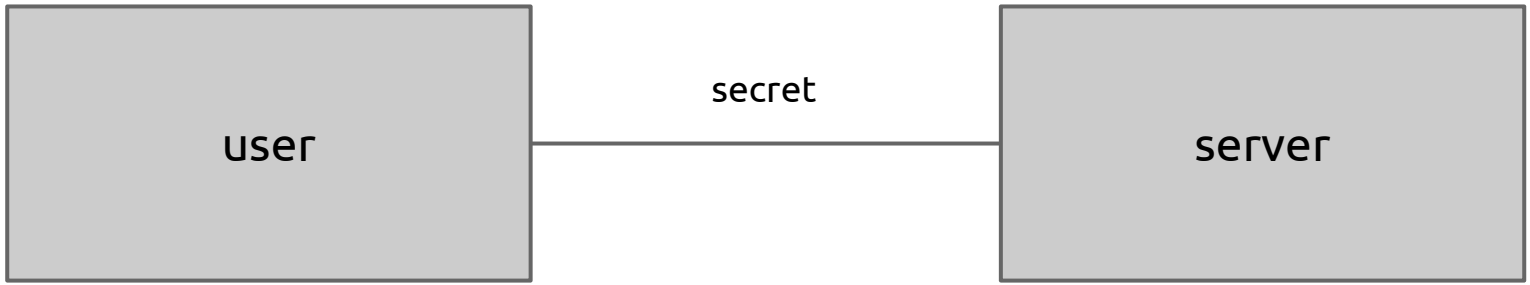
```
$ npm install csurf
```

```
// app.js
var csrf = require('csurf');

app.use(csrf());

app.get('/register', function(req, res) {
  res.render('register.jade', { csrfToken: req.csrfToken() });
});

app.get('/login', function(req, res) {
  res.render('login.jade', { csrfToken: req.csrfToken() });
});
```

```
// register.jade + login.jade
form(method="post")
  input(type="hidden", name="_csrf", value=csrfToken)
```

0x06 - Security

ALWAYS USE SSL!



Securing Cookies

```
app.use(session({
  cookieName: 'session',
  secret: 'some_random_string',
  duration: 30 * 60 * 1000,
  activeDuration: 5 * 60 * 1000,
  httpOnly: true,   // don't let JS code access cookies
  secure: true,     // only set cookies over https
  ephemeral: true,  // destroy cookies when the browser closes
}));
```

0x06 - Other Options

passport.js

Pros

- open source
- supports many different types of login
- very minimalistic

Cons

- requires work to integrate
- mixing multiple authentication types is problematic

drywall

Pros

- open source
- ‘full website framework’
- uses passport.js
- lots of prebuilt stuff!

Cons

- restrictive
- forces you to use specific tools
- doesn’t support API auth (afaik)

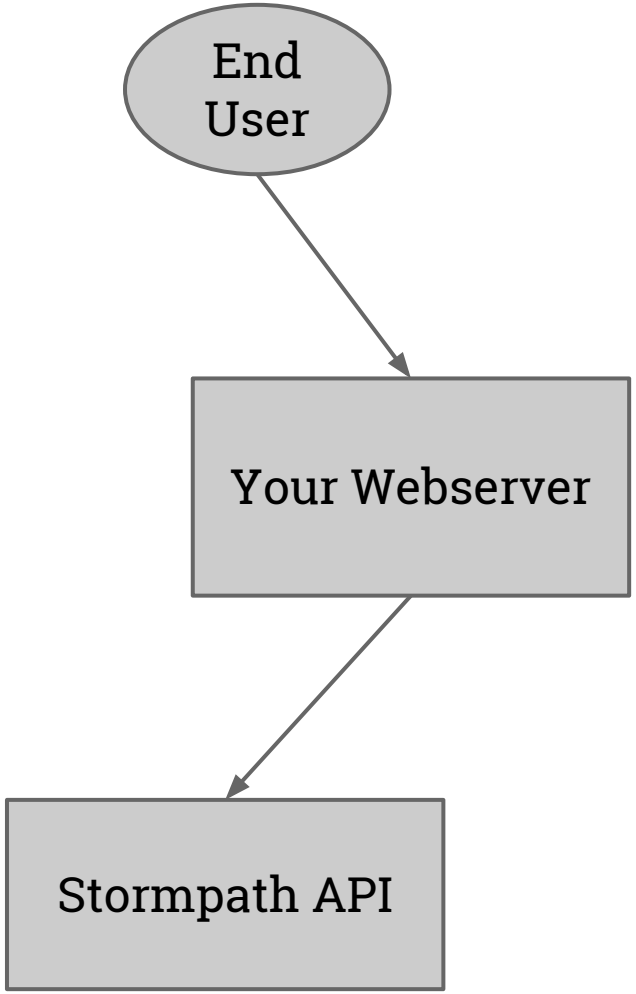
Stormpath

Pros

- free *and* paid versions
- supports both web and api auth
- works in many different languages
- pre-built authentication views
- handles security / storage for you

Cons

- core product is closed source



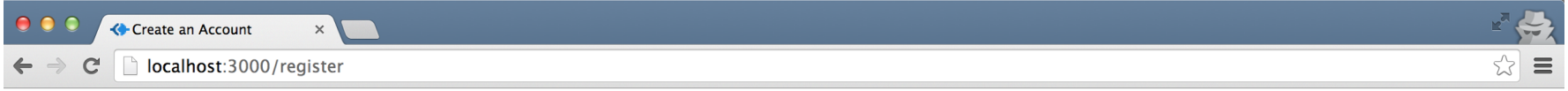
Stormpath

- User account storage / encryption.
- Authentication.
- Authorization.
- REST API management.
- Social login.

express-stormpath

```
$ npm install express-stormpath
```

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

var app = express();
app.use(stormpath.init(app, {
    apiKeyId: 'xxx',
    apiKeySecret: 'xxx',
    application: 'https://api.stormpath.com/v1/applications/xxx',
    secretKey: 'some_long_random_string',
}));

app.listen(3000);
```



Create Account

First Name — First Name

Last Name — Last Name

Email — Email

Password — Password

Create Account

Back to Log In

Enabling Features

```
app.use(stormpath.init(app, {
    apiKeyId: 'xxx',
    apiKeySecret: 'xxx',
    application: 'https://api.stormpath.com/v1/applications/xxx',
    secretKey: 'some_long_random_string',
    enableAccountVerification: true,  // make users confirm their email
    enableForgotPassword: true,       // enable secure password reset
    enableGoogleLogin: true,          // enable google login
}));
```

You're *awesome*.

@rdegges
@gostormpath