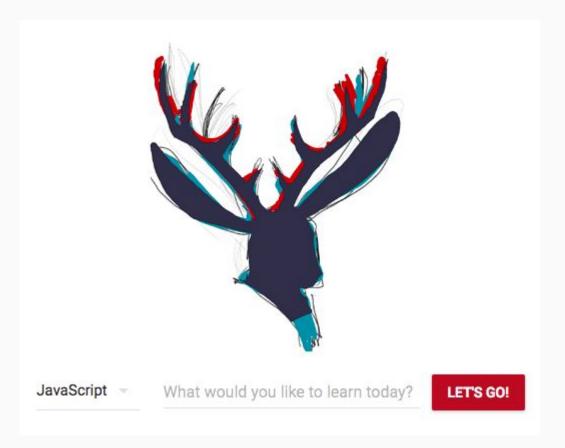
# hello, hackalope.

Created with love by Carla Clay, Tyler Holzer, James Kip, and Daniel Ricaud

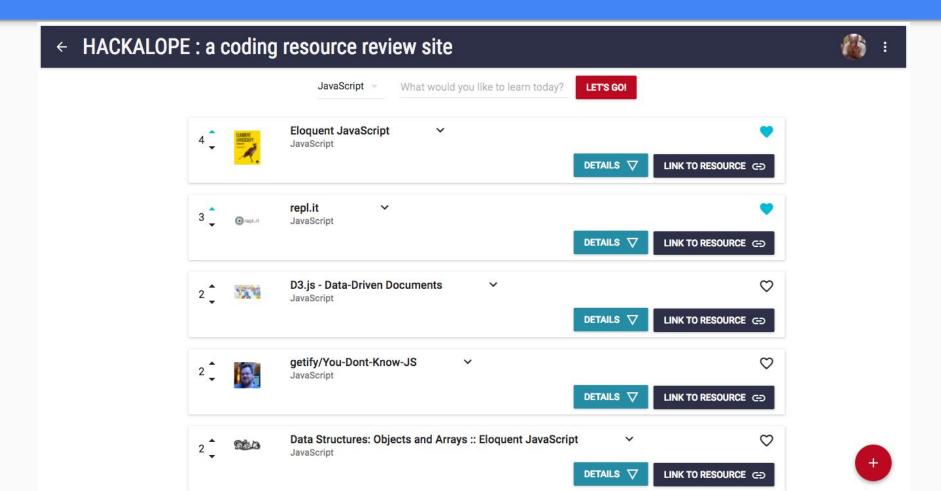
## What would you like to learn today?

Hackalope is a place where programmers can share, search, vote, and comment on coding resources.

Hackalope is inspired by sites like reddit and Yelp, which allow the user base to decide on the quality and visibility of a resource.



#### General search results, with user votes and favorites



## Hackalope's Technologies

React/Redux Express MongoDB Node.js

#### Other technologies:

- Redux Persist
- React Router
- Material UI
- Passport.js
- Mongoose
- Moment.js
- Mocha/Chai/Supertest for TDD
- Webpack
- AWS EC2 for Deployment

#### Detail of a resource, with user comment

#### ← HACKALOPE : a coding resource review site









#### Hackalope's Node.js/Express Server

```
const auth = require('./routes/auth.is');
   const main = require('./routes/main.js');
                                                              app.use('/public', express.static(path.join(__dirname, '/../client/')));
   const search = require('./routes/search.js');
                                                              app.use('/bundle', express.static(path.join( dirname, '/../dist')));
   const results = require('./routes/results.js');
                                                         49
   const submit = require('./routes/submit.js');
                                                         50
   const profile = require('./routes/profile.js');
                                                              app.use('/auth', auth(passport));
   const comments = require('./routes/comments.js');
   const admin = require('./routes/admin.js');
                                                              app.use('/results', results);
   const votes = require('./routes/votes.js');
                                                              app.use('/submit', submit);
                                                              app.use('/profile', profile);
                                                         54
   // USE BLUEBIRD FOR PROMISES
                                                              app.use('/comments', comments);
   mongoose.Promise = require('bluebird');
                                                              app.use('/admin', admin);
                                                              app.use('/votes', votes);
                                                         57
   // START EXPRESS SERVER AND MONGODB
   const app = express();
                                                              app.use('/search', search);
   mongoose.connect('mongodb://localhost/hackalope');
                                                              app.use('/', main);
                                                         60
29
                                                              // CATCH 404 ERRORS
   app.use(bodyParser.urlencoded({ extended: true }));
                                                              app.use((reg, res, next) => {
   app.use(bodyParser.json());
                                                                const err = new Error('Sorry--we couldn\'t find that!');
   app.use(morgan('combined'));
                                                         64
                                                                err.status = 404:
   // PASSPORT AUTHENTICATION
   app.use(session({
                                                         66
     secret: 'hackalope-secret-key',
     resave: false.
                                                         68
     saveUninitialized: false
                                                              const port = process.env.port || 1337;
   app.use(passport.initialize());
                                                              app.listen(port, () => {
   app.use(passport.session());
                                                                console.log(`The server is listening on port: ${port}.`);
                                                         73
   require('./passport/init.js')(passport);
                                                              module.exports = app:
```

#### Hackalope's Redux Reducers and Store

```
import { compose, createStore } from 'redux';
14
                                                             import { persistStore, autoRehydrate } from 'redux-persist';
15
     // COMBINE ALL REDUCERS
                                                             import { localStorage } from 'redux-persist/storages';
     const appReducer = combineReducers({
                                                             import rootReducer from './reducers/rootReducer.jsx';
       results: resultsReducer,
17
                                                             // initialize store and initialize redux dev tools
       result: activeResult,
18
                                                             // this is done as a promise to enable
19
       user: authReducer,
                                                             // authenticaion check when app component loads
20
       form: formReducer,
       submission: submissionReducer.
                                                             export default function configureStore() {
                                                               return new Promise((resolve, reject) => {
       search: searchReducer,
                                                                 try {
23
       dialogs: dialogReducer,
                                                                   const store = createStore(
24
       comments: commentsReducer,
                                                                     rootReducer,
                                                                     undefined,
25
       comment: commentReducer.
26
       unapproved: unapprovedReducer,
                                                                       autoRehydrate(),
       profile: profileReducer,
                                                                      window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__(),
28
       snackbar: snackbarReducer.
29
30
                                                                   persistStore(
     const rootReducer = (state, action) => {
                                                                     { storage: localStorage },
32
       if (action.type === 'LOGOUT') {
                                                                     () => resolve(store),
33
         state = undefined:
34
                                                                   window. REDUX DEVTOOLS EXTENSION && window. REDUX DEVTOOLS EXTENSION ();
       return appReducer(state, action);
35
                                                                 } catch (e) {
36
                                                               });
     export default rootReducer;
                                                         32 };
```

# Hackalope's frontend routes and app component

26

28

30

37

40

50

Hackalope is a single-page application that uses React Router for navigation. It also uses Redux Persist to enable page refreshing.

Persistence was both an asset and a challenge. We set up the Redux store as a promise and required the app component to first check the server for authentication before rendering.

```
const routes = (
    <Route path="/" component={Landing} />
    <Route path="/main" component={Main}>
      <Route path="/main/results" component={ResultsList} />
      <Route path="/login" component={Login} />
      <Route path="/submit" component={Submit} />
    </Route>
    <Route path="/resource/:id" component={ResourceContainer} />
    <Route path="/user" component={User}>
      <Route path="/user/profile" component={Profile} />
      <Route path="/user/admin" component={Admin} />
    </Route>
class App extends React.Component {
  componentDidMount() {
    this.props.dispatch(checkAuth({ checkingAuth: true }));
    findUser(this.props.user, this.props.dispatch);
  render() {
    return this.props.user.checkingAuth ? null : (
      <Router history={hashHistory}>
        {routes}
      </Router>
const mapStateToProps = state => ({
  user: state.user,
});
export default connect(mapStateToProps)(App);
```

```
exports.newVote = function (vote) {
      return VoteModel.create(vote);
    exports.getVotesByUser = function (userId) {
      return VoteModel.find({ user: userId });
    exports.deleteVote = function (resourceId, userId) {
      return VoteModel.remove({ user: userId, resource: resourceId });
16
    exports.updateVote = function (resourceId, userId, newVote) {
      return VoteModel.findOneAndUpdate({ resource: resourceId, user: userId },
        { $set: { vote: newVote } });
22
    exports.getUserVotesForResources = function (array, userId) {
25
      return VoteModel.find({ resource: { $in: array }, user: userId });
26
    exports.getVote = function (resourceId, userId) {
30
      return VoteModel.findOne({ user: userId, resource: resourceId });
33
```

const VoteModel = require('../models/vote.js');

Hackalope employs MongoDB to store user, resource, comment, and vote data.

We developed a library of Mongoose queries to make data retrieval, deletion, and modification modular and clear, and to keep our server routes clean and easy to read.

### Hackalope + ES6

Hackalope fully embraces ES6.

```
    arrow functions
```

- destructuring
- template strings
- let / const
- promises
- modules
- default parameters

Hackalope was linted using the Airbnb style guide.

```
// handles user click to vote on a resource
    export const handleVote = (resourceId, votes, newVote, dispatch) => {
      dispatch(actions.updateVote(resourceId, votes, newVote));
      axios.post(`/votes/${resourceId}`, newVote
        .then((response) => {
          const updatedResource = response.data;
          dispatch(actions.updateResource(updatedResource));
        .catch((err) => {
          console.error(err):
14
    export const isUpvoted = (user, result, votes) => {
      let upvoted = false;
20
      if (user. id & votes) {
        votes.forEach((vote) => {
          if (vote.resource === result._id && vote.vote === 1) {
23
            upvoted = true;
      return upvoted;
29
    // return bool for status of downvote button
    export const isDownvoted = (user, result, votes) => {
      let downvoted = false;
33
      if (user._id && votes) {
34
        votes.forEach((vote) => {
          if (vote.resource === result._id && vote.vote === -1) {
            downvoted = true;
38
40
      return downvoted:
```

## Tyler Holzer

Email: tyler.s.holzer@gmail.com

Phone: 605.415.3537

Website: tylerholzer.com

LinkedIn: linkedin.com/in/tyler-holzer

GitHub: github.com/rezloh