

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.



JavaScript ▼

What would you like to learn today?

LET'S GO!

General search results, with user votes and favorites

← HACKALOPE : a coding resource review site



JavaScript ▾

What would you like to learn today?

LET'S GO!

4
▴
▾



Eloquent JavaScript

JavaScript ▾



DETAILS ▾

LINK TO RESOURCE ↗

3
▴
▾



repl.it

JavaScript ▾



DETAILS ▾

LINK TO RESOURCE ↗

2
▴
▾



D3.js - Data-Driven Documents

JavaScript ▾



DETAILS ▾

LINK TO RESOURCE ↗

2
▴
▾



getify/You-Dont-Know-JS

JavaScript ▾



DETAILS ▾

LINK TO RESOURCE ↗

2
▴
▾



Data Structures: Objects and Arrays :: Eloquent JavaScript

JavaScript ▾



DETAILS ▾

LINK TO RESOURCE ↗



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



getify/You-Dont-Know-JS
JavaScript

LINK TO RESOURCE ↗

Site Description:

You-Dont-Know-JS - A book series on JavaScript. @YDKJS on twitter.

What's it about?

Github repo for the popular javascript book "You don't know JS"

Books

Free

Github

Repo

Javascript

Add a comment

COMMENT



05/02/2017, 1:26 pm

Kyle Simpson is a rock star! Really deep resource for learning JS.



Hackalope's Node.js/Express Server

```
11 // ROUTES
12 const auth = require('./routes/auth.js');
13 const main = require('./routes/main.js');
14 const search = require('./routes/search.js');
15 const results = require('./routes/results.js');
16 const submit = require('./routes/submit.js');
17 const profile = require('./routes/profile.js');
18 const comments = require('./routes/comments.js');
19 const admin = require('./routes/admin.js');
20 const votes = require('./routes/votes.js');
21
22 // USE BLUEBIRD FOR PROMISES
23 mongoose.Promise = require('bluebird');
24
25 // START EXPRESS SERVER AND MONGODB
26 const app = express();
27 mongoose.connect('mongodb://localhost/hackalope');
28
29 // MIDDLEWARE
30 app.use(bodyParser.urlencoded({ extended: true }));
31 app.use(bodyParser.json());
32 app.use(morgan('combined'));
33
34 // PASSPORT AUTHENTICATION
35 app.use(session({
36   secret: 'hackalope-secret-key',
37   resave: false,
38   saveUninitialized: false
39 }));
40 app.use(passport.initialize());
41 app.use(passport.session());
42
43 // INITIALIZE SERIALIZE/DESERIALIZE FUNCTIONS
44 require('./passport/init.js')(passport);
45
46 // SERVE STATIC FILES
47 app.use('/public', express.static(path.join(__dirname, '../client/')));
48 app.use('/bundle', express.static(path.join(__dirname, '../dist')));
49
50 // ROUTING
51 app.use('/auth', auth(passport));
52 app.use('/results', results);
53 app.use('/submit', submit);
54 app.use('/profile', profile);
55 app.use('/comments', comments);
56 app.use('/admin', admin);
57 app.use('/votes', votes);
58 app.use('/search', search);
59 app.use('/', main);
60
61 // CATCH 404 ERRORS
62 app.use((req, res, next) => {
63   const err = new Error('Sorry--we couldn\'t find that!');
64   err.status = 404;
65   next(err);
66 });
67
68 // LISTEN
69 const port = process.env.port || 1337;
70 app.listen(port, () => {
71   console.log(`The server is listening on port: ${port}.`);
72 });
73
74 module.exports = app;
```

Hackalope's Redux Reducers and Store

```
14
15 // COMBINE ALL REDUCERS
16 const appReducer = combineReducers({
17   results: resultsReducer,
18   result: activeResult,
19   user: authReducer,
20   form: formReducer,
21   submission: submissionReducer,
22   search: searchReducer,
23   dialogs: dialogReducer,
24   comments: commentsReducer,
25   comment: commentReducer,
26   unapproved: unapprovedReducer,
27   profile: profileReducer,
28   snackbar: snackbarReducer,
29 });
30
31 const rootReducer = (state, action) => {
32   if (action.type === 'LOGOUT') {
33     state = undefined;
34   }
35   return appReducer(state, action);
36 };
37
38 export default rootReducer;
```

```
1 import { compose, createStore } from 'redux';
2 import { persistStore, autoRehydrate } from 'redux-persist';
3 import { localStorage } from 'redux-persist/storages';
4 import rootReducer from './reducers/rootReducer.jsx';
5
6 // initialize store and initialize redux dev tools
7 // this is done as a promise to enable
8 // authentication check when app component loads
9
10 export default function configureStore() {
11   return new Promise((resolve, reject) => {
12     try {
13       const store = createStore(
14         rootReducer,
15         undefined,
16         compose(
17           autoRehydrate(),
18           window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__(),
19         ),
20       );
21
22       persistStore(
23         store,
24         { storage: localStorage },
25         () => resolve(store),
26       );
27       window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__();
28     } catch (e) {
29       reject(e);
30     }
31   });
32 };
```


Hackalope's frontend routes and app component

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.

```
21 const routes = (  
22   <div>  
23     <Route path="/" component={Landing} />  
24     <Route path="/main" component={Main}>  
25       <Route path="/main/results" component={ResultsList} />  
26       <Route path="/login" component={Login} />  
27       <Route path="/submit" component={Submit} />  
28     </Route>  
29     <Route path="/resource/:id" component={ResourceContainer} />  
30     <Route path="/user" component={User}>  
31       <Route path="/user/profile" component={Profile} />  
32       <Route path="/user/admin" component={Admin} />  
33     </Route>  
34   </div>  
35 );  
36  
37 class App extends React.Component {  
38  
39   componentDidMount() {  
40     this.props.dispatch(checkAuth({ checkingAuth: true }));  
41     findUser(this.props.user, this.props.dispatch);  
42   }  
43  
44   render() {  
45     return this.props.user.checkingAuth ? null : (  
46       <Router history={hashHistory}>  
47         {routes}  
48       </Router>  
49     );  
50   }  
51 }  
52  
53 const mapStateToProps = state => ({  
54   user: state.user,  
55 });  
56  
57 export default connect(mapStateToProps)(App);
```



```

1  const VoteModel = require('../models/vote.js');
2
3  // creates a new vote for a resource if none exists
4  exports.newVote = function (vote) {
5    return VoteModel.create(vote);
6  };
7
8  // gets all of a user's votes
9  exports.getVotesByUser = function (userId) {
10    return VoteModel.find({ user: userId });
11  };
12
13  // deletes a user's vote on a resource
14  exports.deleteVote = function (resourceId, userId) {
15    return VoteModel.remove({ user: userId, resource: resourceId });
16  };
17
18  // changes a user's vote on a resource
19  exports.updateVote = function (resourceId, userId, newVote) {
20    return VoteModel.findOneAndUpdate({ resource: resourceId, user: userId },
21      { $set: { vote: newVote } });
22  };
23
24  // retrieves user's votes for the a set of resources resources
25  exports.getUserVotesForResources = function (array, userId) {
26    return VoteModel.find({ resource: { $in: array }, user: userId });
27  };
28
29  // retrieves a user's vote on a specific resource
30  exports.getVote = function (resourceId, userId) {
31    return VoteModel.findOne({ user: userId, resource: resourceId });
32  };
33
34
35
36

```

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.

```
4 // handles user click to vote on a resource
5 export const handleVote = (resourceId, votes, newVote, dispatch) => {
6   dispatch(actions.updateVote(resourceId, votes, newVote));
7   axios.post(`/votes/${resourceId}`, newVote)
8     .then((response) => {
9       const updatedResource = response.data;
10      dispatch(actions.updateResource(updatedResource));
11    })
12    .catch((err) => {
13      console.error(err);
14    });
15 };
16
17 // return bool for status of upvote button
18 export const isUpvoted = (user, result, votes) => {
19   let upvoted = false;
20   if (user._id && votes) {
21     votes.forEach((vote) => {
22       if (vote.resource === result._id && vote.vote === 1) {
23         upvoted = true;
24       }
25     });
26   }
27   return upvoted;
28 };
29
30 // return bool for status of downvote button
31 export const isDownvoted = (user, result, votes) => {
32   let downvoted = false;
33   if (user._id && votes) {
34     votes.forEach((vote) => {
35       if (vote.resource === result._id && vote.vote === -1) {
36         downvoted = true;
37       }
38     });
39   }
40   return downvoted;
41 }
```

Tyler Holzer

Email: tyler.s.holzer@gmail.com

Phone: 605.415.3537

Website: tylerholzer.com

LinkedIn: [linkedin.com/in/tyler-holzer](https://www.linkedin.com/in/tyler-holzer)

GitHub: github.com/rezloh

