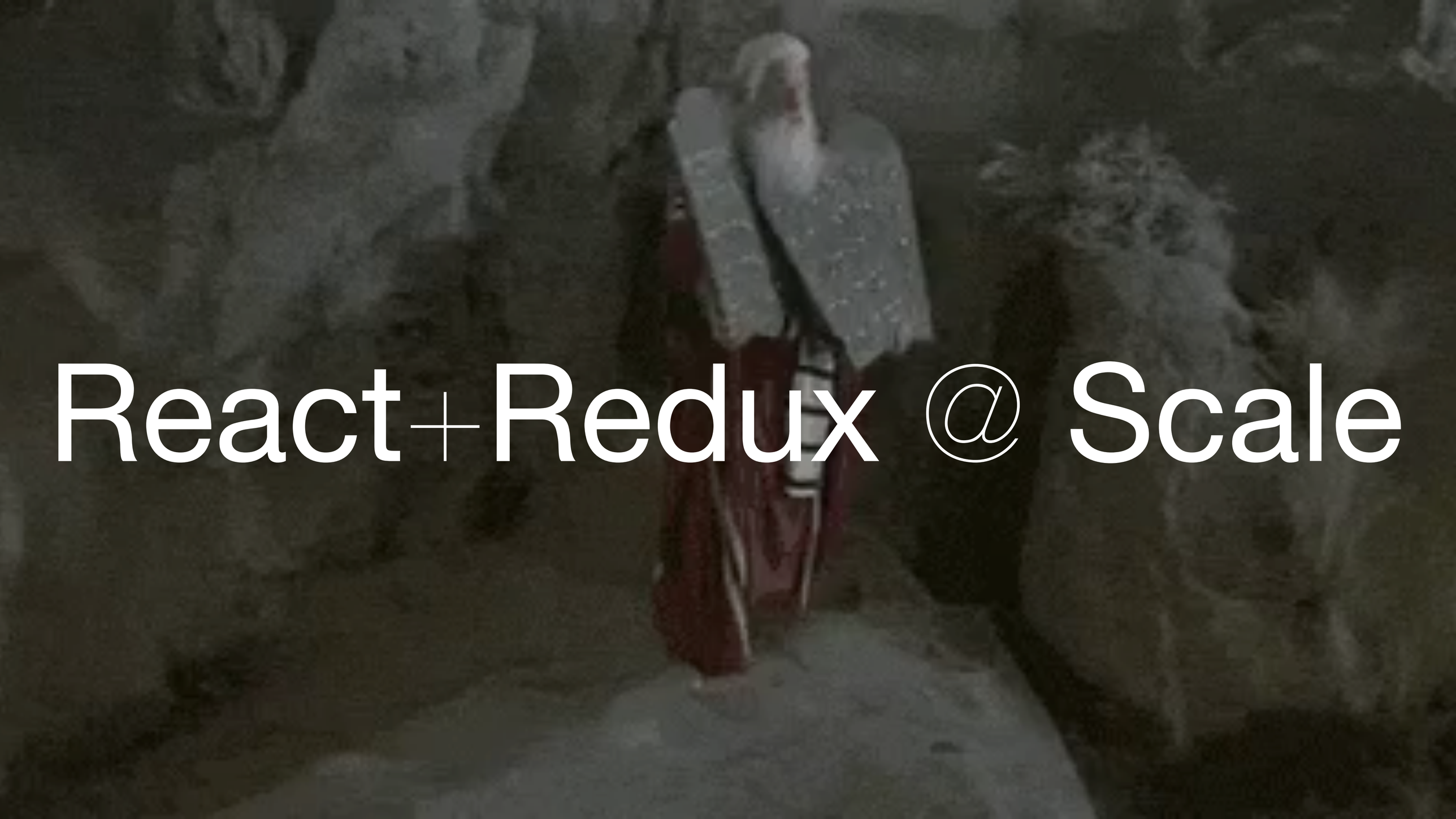

React+Redux @ Scale


@dcousineau


QUEENS JS
NEW YORK
NEW YORK
QUEENS JS

eact

mple
rce

ct + Backbone.js
no, Source

a.js + React
no, Source

eScript + React
no, Source

ct + Alt
no, Source

---

*act is a JavaScript library for creating
er interfaces. Its core principles are
clarative code, efficiency, and
xibility. Simply specify what your
mponent looks like and React will
ep it up-to-date when the underlying
ta changes.*

"

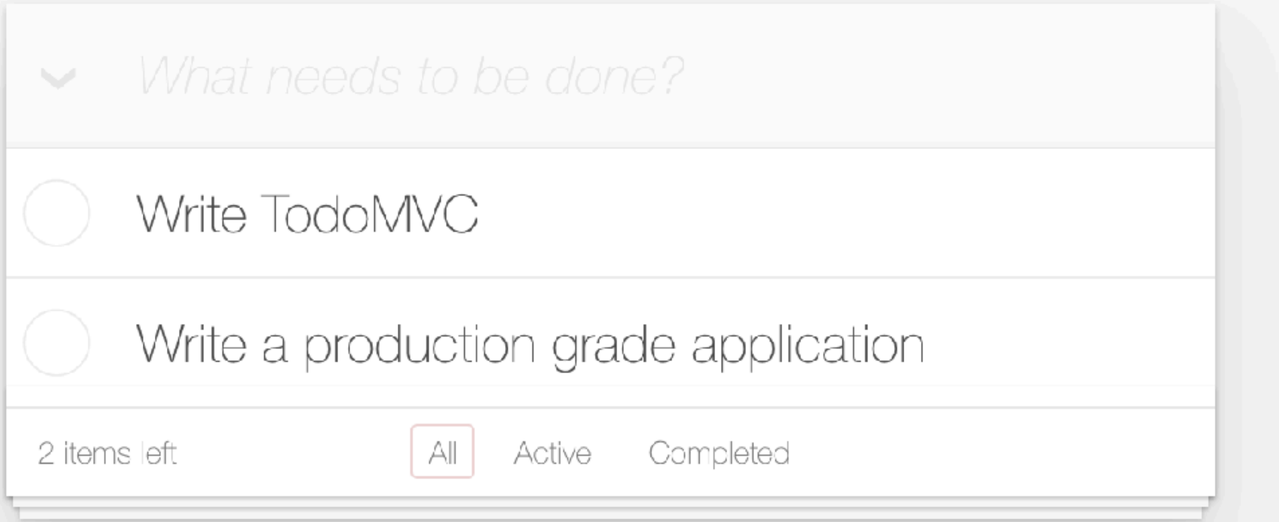
React

---

icial Resources

Tutorial
Philosophy
Support

todos



Double click to edit a todo

Created by petehunt

Part of TodoMVC

eact

mple
rce

ct + Backbone.js
o, Source

a.js + React
o, Source

eScript + React
o, Source

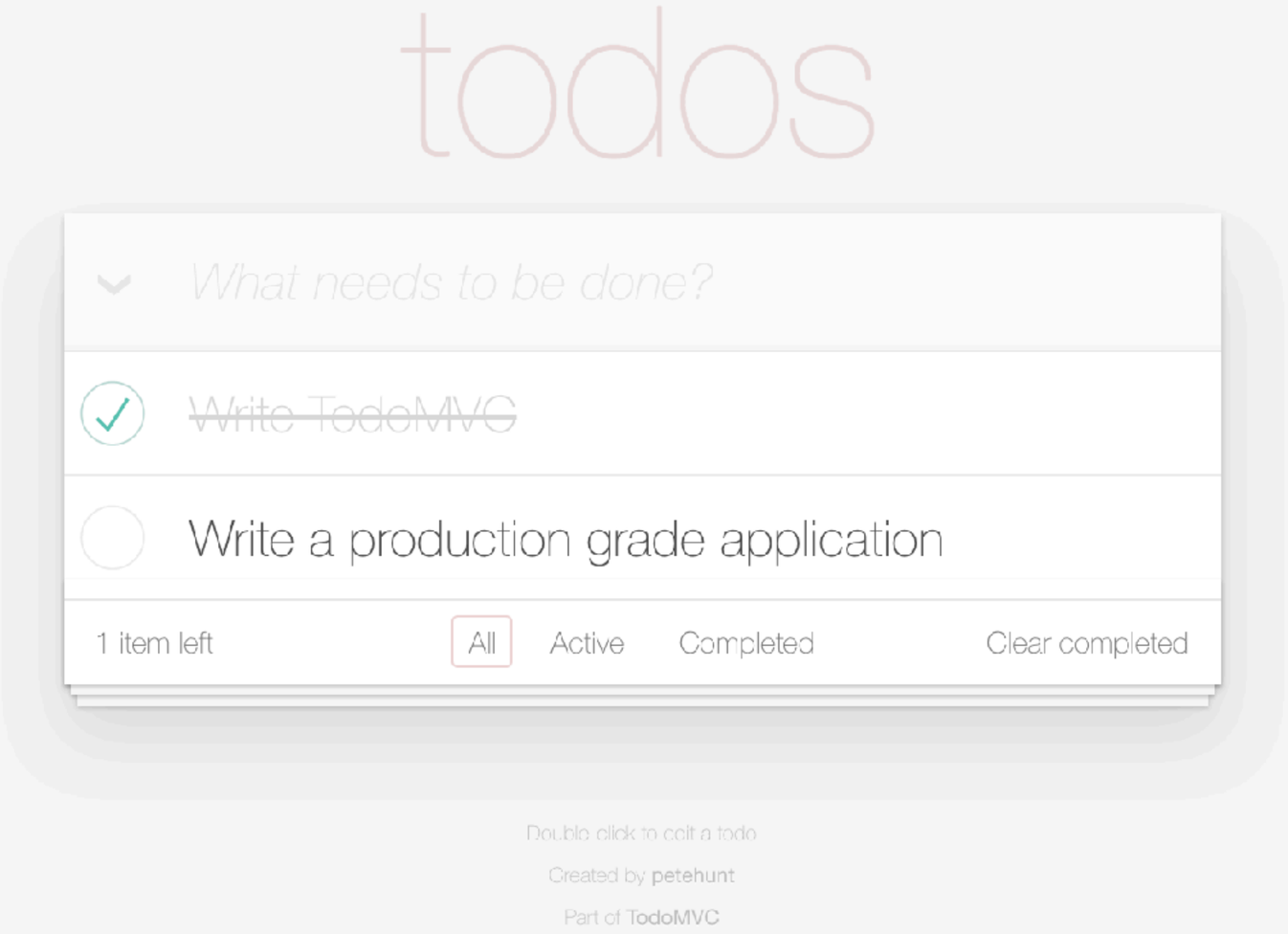
ct + Alt
o, Source

act is a JavaScript library for creating
er interfaces. Its core principles are
clarative code, efficiency, and
ibility. Simply specify what your
mponent looks like and React will
ep it up-to-date when the underlying
ta changes.

React

icial Resources

Tutorial
Philosophy
Support



How to draw an Owl.

"A fun and creative guide for beginners"

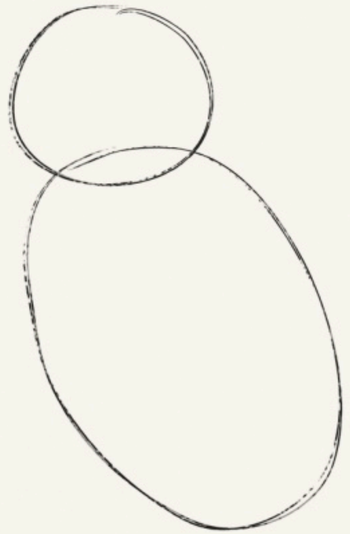


Fig 1. Draw two circles

Fig 2. Draw the rest of the damn Owl

Rules

“Rules”


THE
REAL
WORLD

Scalability is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth.

– Wikipedia

Part 1: React

Rule: Components should be stateless

Reality: State is the enemy, but also inevitable

```
onClick(e) {
    const value = e.target.value;
    const formatted = value.toUpperCase();
    this.setState({value: formatted});
}
```

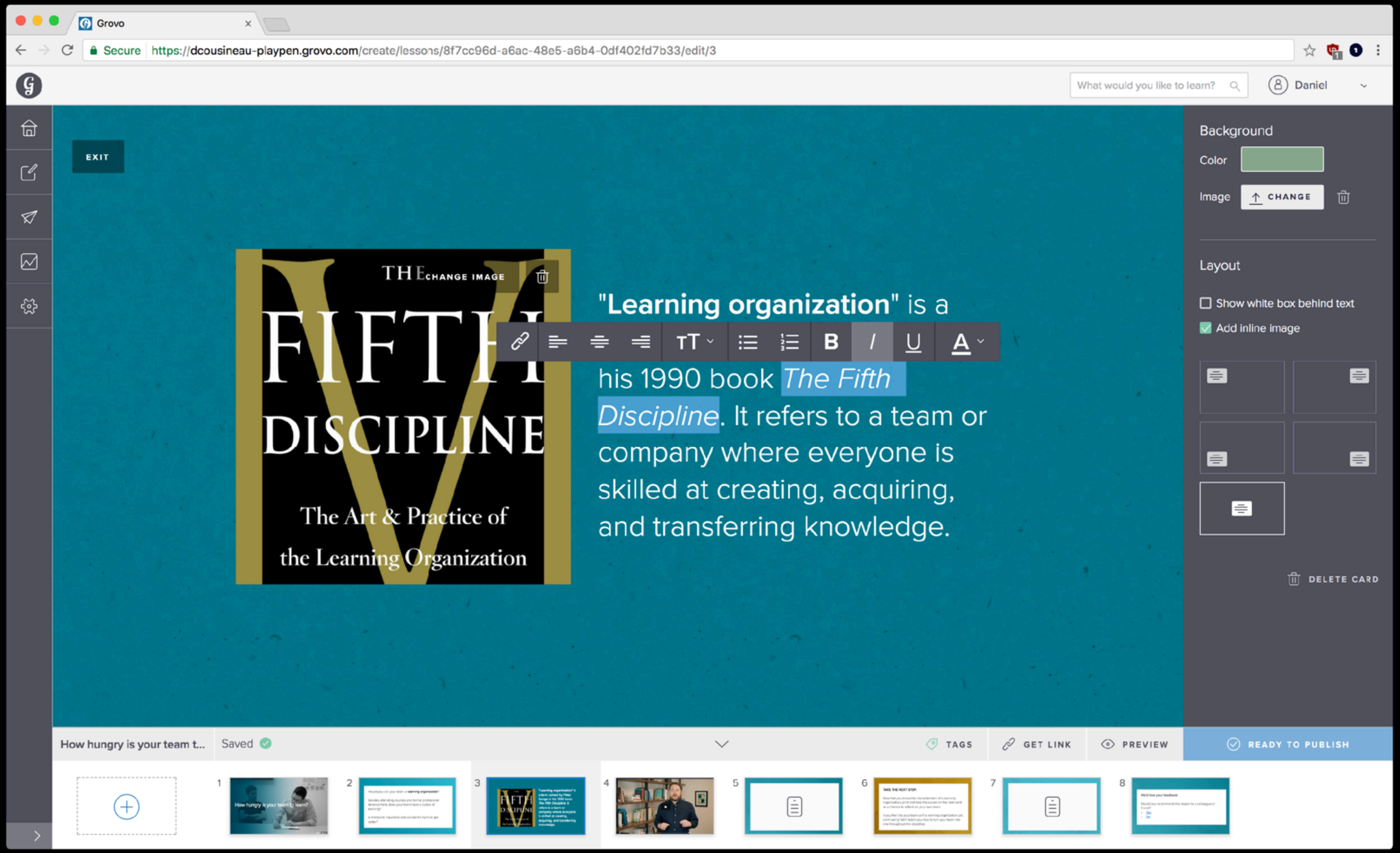
```
onClick() {
    this.setState((previousState, currentProps) => {
        return {
            show: !previousState.show,
        };
    });
}
```

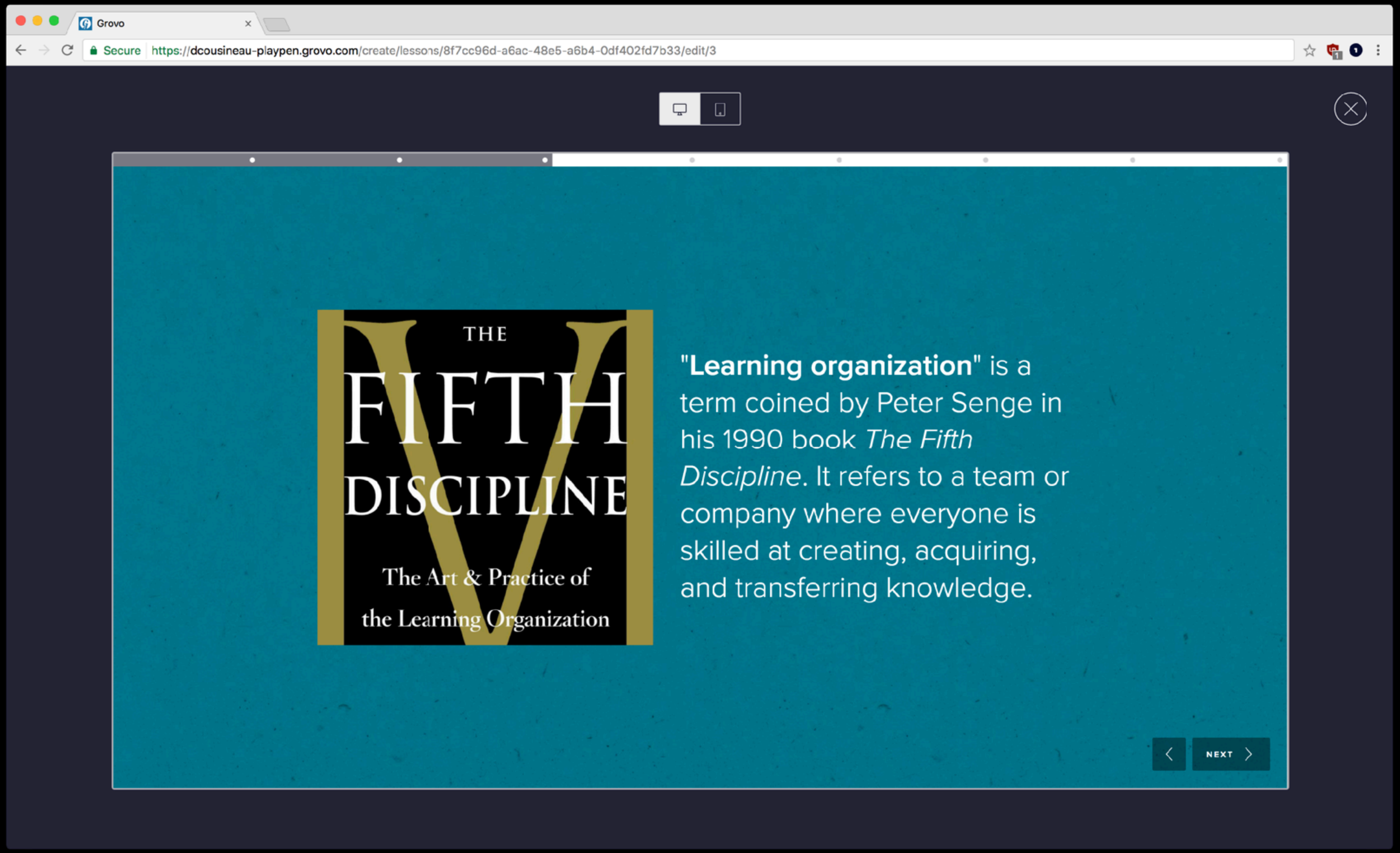

onClick(e) {
this.setState
e.target.value});
this.props.on
his.state.value);
}

```
onClick(e) {
    this.setState({value: e.target.value}, () => {
        this.props.onChange(this.state.value);
    });
}
```

Rule: Don’t use Context, it hides complexity

Reality: Sometimes complexity should be hidden


Grovo
Secure
https://dcousineau-playpen.grovo.com/create/lessons/8f7cc96d-a6ac-48e5-a6b4-0df402fd7b33/edit/3
What would you like to learn?
Daniel
EXIT
THE CHANGE IMAGE
FIFTH DISCIPLINE
The Art & Practice of the Learning Organization
"Learning organization" is a
his 1990 book The Fifth Discipline. It refers to a team or company where everyone is skilled at creating, acquiring, and transferring knowledge.
Background
Color
Image
CHANGE
Layout
Show white box behind text
Add inline image
DELETE CARD
How hungry is your team t...
Saved
TAGS
GET LINK
PREVIEW
READY TO PUBLISH


Grovo
Secure
https://dcousineau-playpen.grovo.com/create/lessons/8f7cc96d-a6ac-48e5-a6b4-0df402fd7b33/edit/3
THE
FIFTH
DISCIPLINE
The Art & Practice of
the Learning Organization
"Learning organization" is a term coined by Peter Senge in his 1990 book *The Fifth Discipline*. It refers to a team or company where everyone is skilled at creating, acquiring, and transferring knowledge.
NEXT

```
class TextCard extends React.Component {
    static contextTypes = {
        metatypes: React.PropTypes.object,
    };

    render() {
        const {cardData} = this.props;
        const {metatypes} = this.context;

        return (
            <div>
                The following is either editable or displayed:
                <metatypes.text value={cardData.text} onChange={this.props.onChange} />
            </div>
        )
    }
}

function selectCardComponent(cardData) {
    switch (cardData.type) {
        case 'text': return TextCard;
        default: throw new Error(`Invalid card type ${cardData.type}`);
    }
}
```

```
class TextCard extends React.Component {
    static contextTypes = {
        metatypes: React.PropTypes.object,
    };

    render() {
        const {cardData} = this.props;
        const {metatypes} = this.context;

        return (
            <div>
                The following is either editable or displayed:
                <metatypes.text value={cardData.text} onChange={this.props.onChange} />
            </div>
        )
    }
}

function selectCardComponent(cardData) {
    switch (cardData.type) {
        case 'text': return TextCard;
        default: throw new Error(`Invalid card type ${cardData.type}`);
    }
}
```

```
const metatypesEdit = {
    text: class extends React.Component {
        render() {
            return <input type="text" {...this.props} />;
        }
    }
}

const metatypesView = {
    text: class extends React.Component {
        render() {
            return <span>{this.props.value}</span>;
        }
    }
}
```

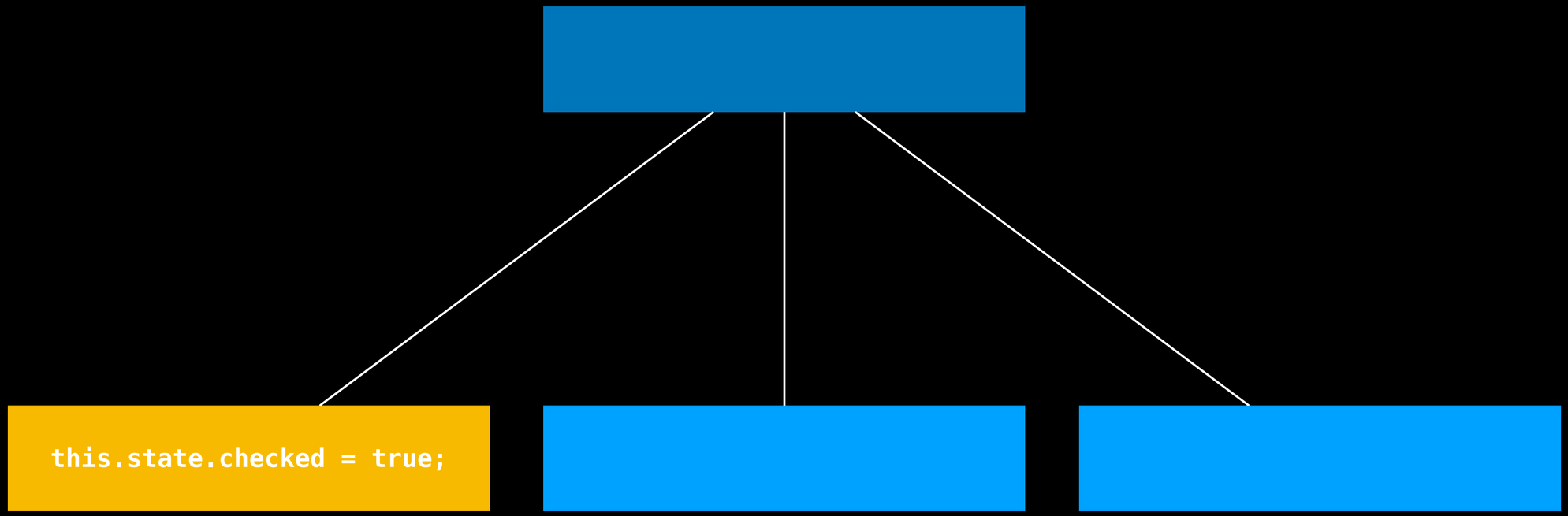
```
class CardViewer extends React.Component {
    static childContextTypes = {
        metatypes: React.PropTypes.object
    };

    getChildContext() {
        return {metatypes: metatypesView};
    }

    render() {
        const {cardData} = this.props;
        const CardComponent = selectCardComponent(cardData);

        return <CardComponent cardData={cardData} />
    }
}
```

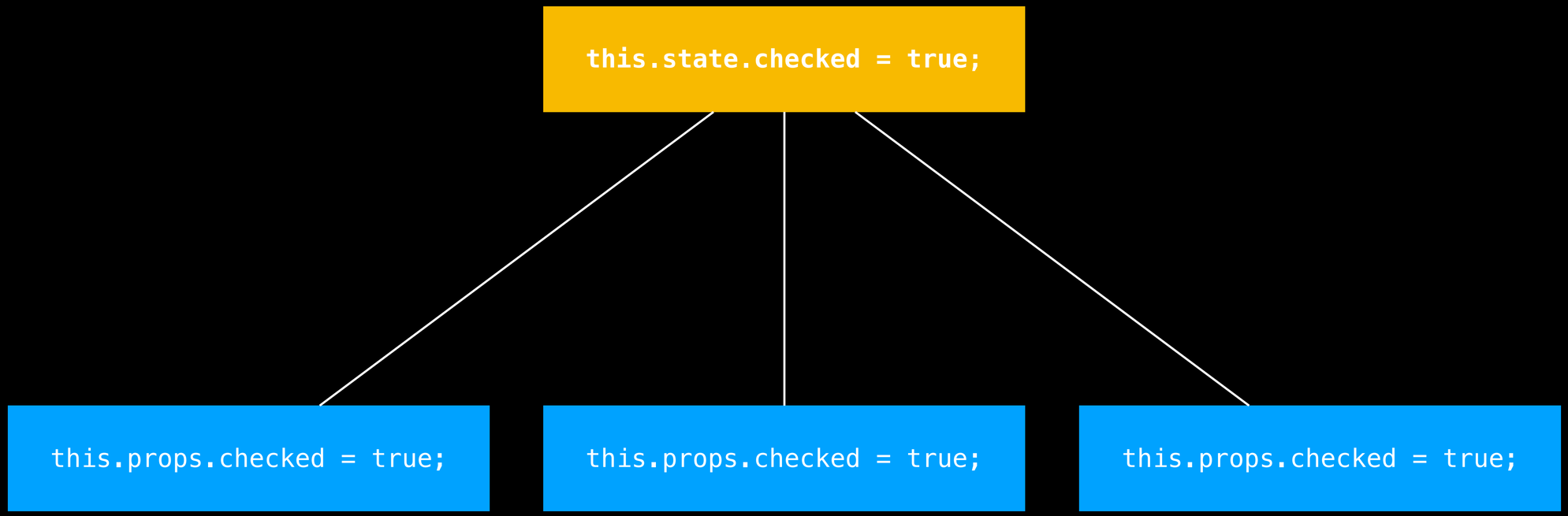
```
class CardEditor extends React.Component {
    static childContextTypes = {
        metatypes: React.PropTypes.object
    };

    getChildContext() {
        return {metatypes: metatypesEdit};
    }

    render() {
        const {cardData} = this.props;
        const CardComponent = selectCardComponent(cardData);

        return <CardComponent cardData={cardData} />
    }
}
```

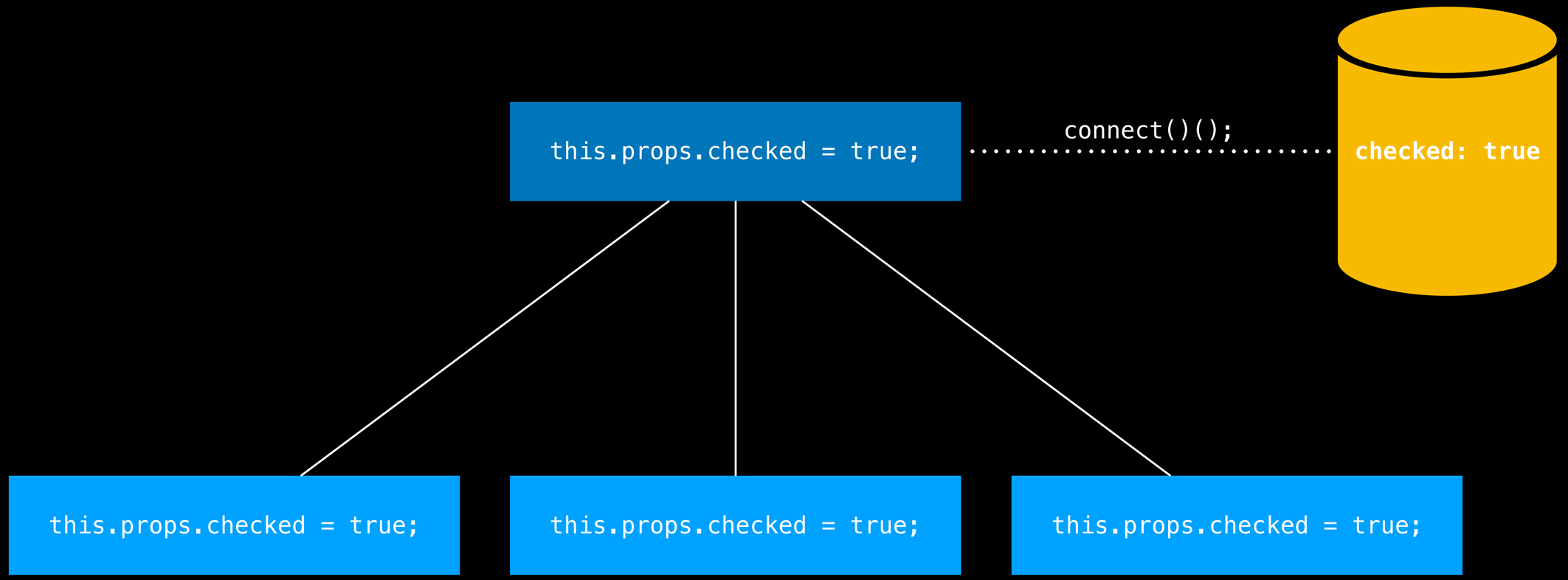
Part 2: Redux

Rule: “Single source of truth” means all state in the store

Reality: You can have multiple “single sources”


this.state.checked = true;


this.state.checked = true;
this.props.checked = true;
this.props.checked = true;
this.props.checked = true;


this.props.checked = true;
connect()();
checked: true
this.props.checked = true;
this.props.checked = true;
this.props.checked = true;

```
window.location.*
```

Rule: Side effects should happen outside the Redux cycle

Reality: This doesn’t mean you can’t have callbacks

```
function persistPostAction(post, callback = () => {}) {
    return {
        type: 'PERSIST_POST',
        post,
        callback
    };
}

function *fetchPostsSaga(action) {
    const status = yield putPostAPI(action.post);
    yield put(persistPostCompleteAction(status));
    yield call(action.callback, status);
}

class ComposePost extends React.Component {
    onClickSubmit() {
        const {dispatch} = this.props;
        const {post} = this.state;
        dispatch(persistPostAction(post, () => this.displaySuccessBanner()));
    }
}
```

```
class ViewPostPage extends React.Component {
    componentWillMount() {
        const {dispatch, postId} = this.props;
        dispatch(fetchPostAction(postId, () => this.logPageLoadComplete()));
    }
}
```

Rule: Redux stores must be normalized for performance

Reality: You must normalize to reduce complexity

https://medium.com/@dcousineau/advanced-redux-entity-normalization-f5f1fe2aefc5

```
{
  byId: {
    ...entities
  },
  keyWindows: [`${keyWindowName}`],
  [keyWindowName]: {
    ids: ['id0', ..., 'idN'],
    ...meta
  }
}
```

```
{
  byId: {
    'a': userA, 'b': userB, 'c': userC, 'd': userD
  },
  keyWindows: ['browseUsers', 'allManagers'],
  browseUsers: {
    ids: ['a', 'b', 'c'],
    isFetching: false,
    page: 1,
    totalPages: 10,
    next: '/users?page=2',
    last: '/users?page=10'
  },
  allManagers: {
    ids: ['d', 'a'],
    isFetching: false
  }
}
```

```
function selectUserById(store, userId) {
  return store.users.byId[userId];
}

function selectUsersByKeyWindow(store, keyWindow) {
  return store.users[keyWindow].ids.map(userId => selectUserById(store, userId));
}
```

```
function fetchUsers({query}, keyWindow) {
  return {
    type: FETCH_USERS,
    query,
    keyWindow
  };
}

function fetchManagers() {
  return fetchUsers({query: {isManager: true}}, 'allManager');
}

function receiveEntities(entities, keyWindow) {
  return {
    type: RECEIVE_ENTITIES,
    entities,
    keyWindow
  };
}
```

```
function reducer(state = defaultState, action) {
  switch(action.type) {
    case FETCH_USERS:
      return {
        ...state,
        keyWindows: uniq([...state.keyWindows, action.keyWindow]),
        [action.keyWindow]: {
          ...state[action.keyWindow],
          isFetching: true,
          query: action.query
        }
      };
    case RECEIVE_ENTITIES:
      return {
        ...state,
        byId: {
          ...state.byId,
          ...action.entities.users.byId
        },
        keyWindows: uniq([...state.keyWindows, action.keyWindow]),
        [action.keyWindow]: {
          ...state[action.keyWindow],
          isFetching: false,
          ids: action.entities.users.ids
        }
      };
  }
}
```

```
function reducer(state = defaultState, action) {
  switch(action.type) {
    case FETCH_USERS:
      return {
        ...state,
        keyWindows: uniq([...state.keyWindows, action.keyWindow]),
        [action.keyWindow]: {
          ...state[action.keyWindow],
          isFetching: true,
          query: action.query
        }
      };
    case RECEIVE_ENTITIES:
      return {
        ...state,
        byId: {
          ...state.byId,
          ...action.entities.users.byId
        },
        keyWindows: uniq([...state.keyWindows, action.keyWindow]),
        [action.keyWindow]: {
          ...state[action.keyWindow],
          isFetching: false,
          ids: action.entities.users.ids
        }
      };
  }
}
```

```
function selectUsersAreFetching(store, keyWindow) {
  return !!store.users[keyWindow].isFetching;
}

function selectManagersAreFetching(store) {
  return selectUsersAreFetching(store, 'allManagers');
}
```

```
function reducer(state = defaultState, action) {
  switch(action.type) {
    case UPDATE_USER:
      return {
        ...state,
        draftsById: {
          ...state.draftsById,
          [action.user.id]: action.user
        }
      };
    case RECEIVE_ENTITIES:
      return {
        ...state,
        byId: {
          ...state.byId,
          ...action.entities.users.byId
        },
        draftsById: {
          ...omit(state.draftsById, action.entities.users.byId)
        },
        keyWindows: uniq([...state.keyWindows, action.keyWindow]),
        [action.keyWindow]: {
          ...state[action.keyWindow],
          isFetching: false,
          ids: action.entities.users.ids
        }
      };
  }
}
```

```
function reducer(state = defaultState, action) {
  switch(action.type) {
    case UPDATE_USER:
      return {
        ...state,
        draftsById: {
          ...state.draftsById,
          [action.user.id]: action.user
        }
      };
    case RECEIVE_ENTITIES:
      return {
        ...state,
        byId: {
          ...state.byId,
          ...action.entities.users.byId
        },
        draftsById: {
          ...omit(state.draftsById, action.entities.users.byId)
        },
        keyWindows: uniq([...state.keyWindows, action.keyWindow]),
        [action.keyWindow]: {
          ...state[action.keyWindow],
          isFetching: false,
          ids: action.entities.users.ids
        }
      };
  }
}
```

```
function selectUserById(store, userId) {
  return store.users.draftsById[userId] || store.users.byId[userId];
}
```

```
function reducer(state = defaultState, action) {
  switch(action.type) {
    case UNDO_UPDATE_USER:
      return {
        ...state,
        draftsById: {
          ...omit(state.draftsById, action.user.id),
        }
      };
  }
}
```

Part 3: Scale

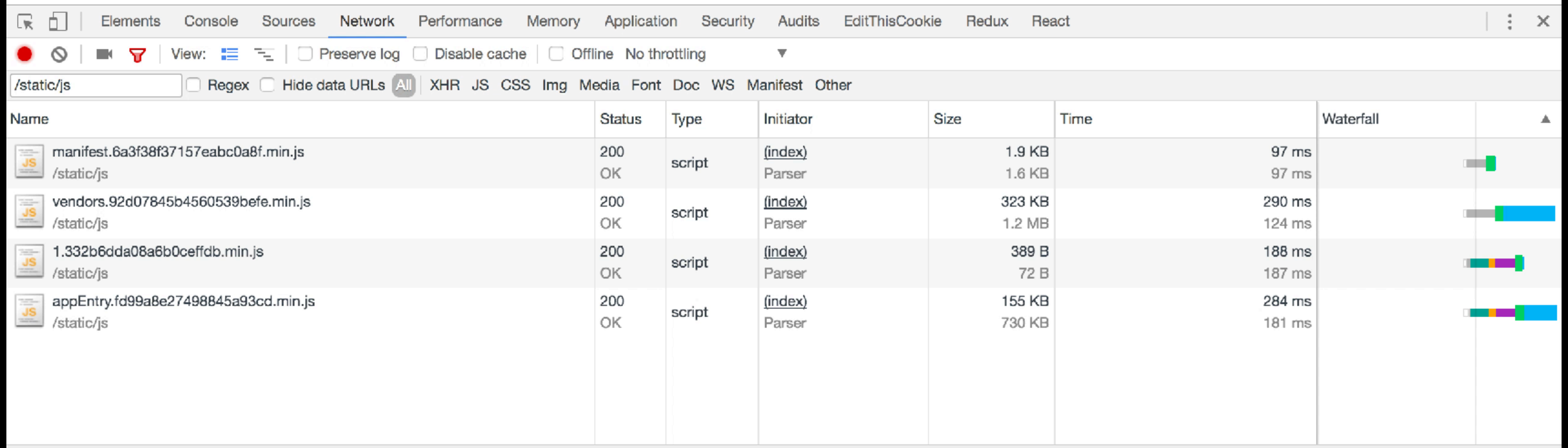
Rule: Keep dependencies low to keep the application fast

Reality: Use bundling to increase PERCEIVED performance

```
class Routes extends React.Component {
  render() {
    return (
      <Switch>
        <Route exact path="/"
               component={require('../home').default} />

        <Route path="/admin"
               component={lazy(require('bundle-loader?lazy&name=admin!../admin'))} />

        <Route component={PageNotFound} />
      </Switch>
    );
  }
}
```

```
require('bundle-loader?lazy&name=admin!../admin’)
```

```
const lazy = loader => class extends React.Component {
    componentWillMount() {
        loader(mod =>
            this.setState({
                Component: mod.default ? mod.default : mod
            })
        );
    }

    render() {
        const { Component } = this.state;

        if (Component !== null) {
            return <Component {...this.props} />;
        } else {
            return <div>Is Loading!</div>;
        }
    }
};
```


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
EditThisCookie
Redux
React
View:
Preserve log
Disable cache
Offline
No throttling
/static/js
Regex
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
Name
Status
Type
Initiator
Size
Time
Waterfall
manifest.6a3f38f37157eabc0a8f.min.js
/static/js
200
OK
script
(index)
Parser
1.9 KB
1.6 KB
97 ms
97 ms
vendors.92d07845b4560539befe.min.js
/static/js
200
OK
script
(index)
Parser
323 KB
1.2 MB
290 ms
124 ms
1.332b6dda08a6b0ceffdb.min.js
/static/js
200
OK
script
(index)
Parser
389 B
72 B
188 ms
187 ms
appEntry.fd99a8e27498845a93cd.min.js
/static/js
200
OK
script
(index)
Parser
155 KB
730 KB
284 ms
181 ms

Rule: Render up-to-date data

Reality: If you got something render it, update it later

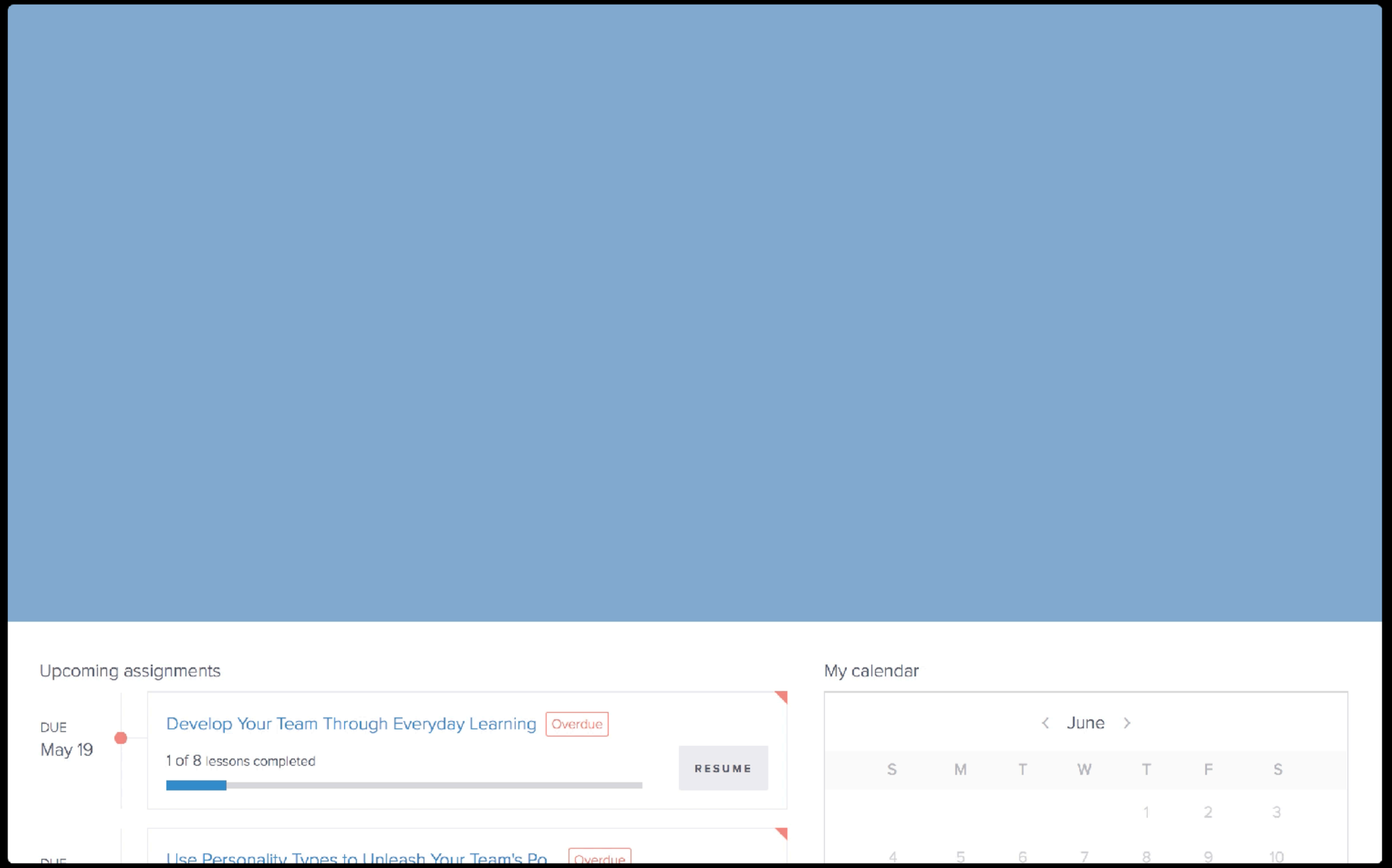
Upcoming assignments

You don't have any upcoming assignments.

My calendar

‹ June ›

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10


Upcoming assignments
DUE
May 19
Develop Your Team Through Everyday Learning
Overdue
1 of 8 lessons completed
RESUME
Use Personality Types to Unleash Your Team's Po
Overdue
My calendar
June
S M T W T F S
1 2 3
4 5 6 7 8 9 10

OVERDUE LESSON:

START

Upcoming assignments

DUE
May 19

Develop Your Team Through Everyday Learning Overdue

1 of 8 lessons completed

RESUME

Use Personality Types to Unleash Your Team's Po Overdue

My calendar

June

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10

Upcoming assignments

DUE
May 19

Develop Your Team Through Everyday Learning Overdue

1 of 8 lessons completed

RESUME

Use Personality Types to Unleash Your Team's Po... Overdue

My calendar

June

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10

Epilog: Scale?

Rule: Scale is bytes served, users concurrent

Reality: Scale is *responding to* bytes served and users concurrent

How fast can you deploy?

Quick Start

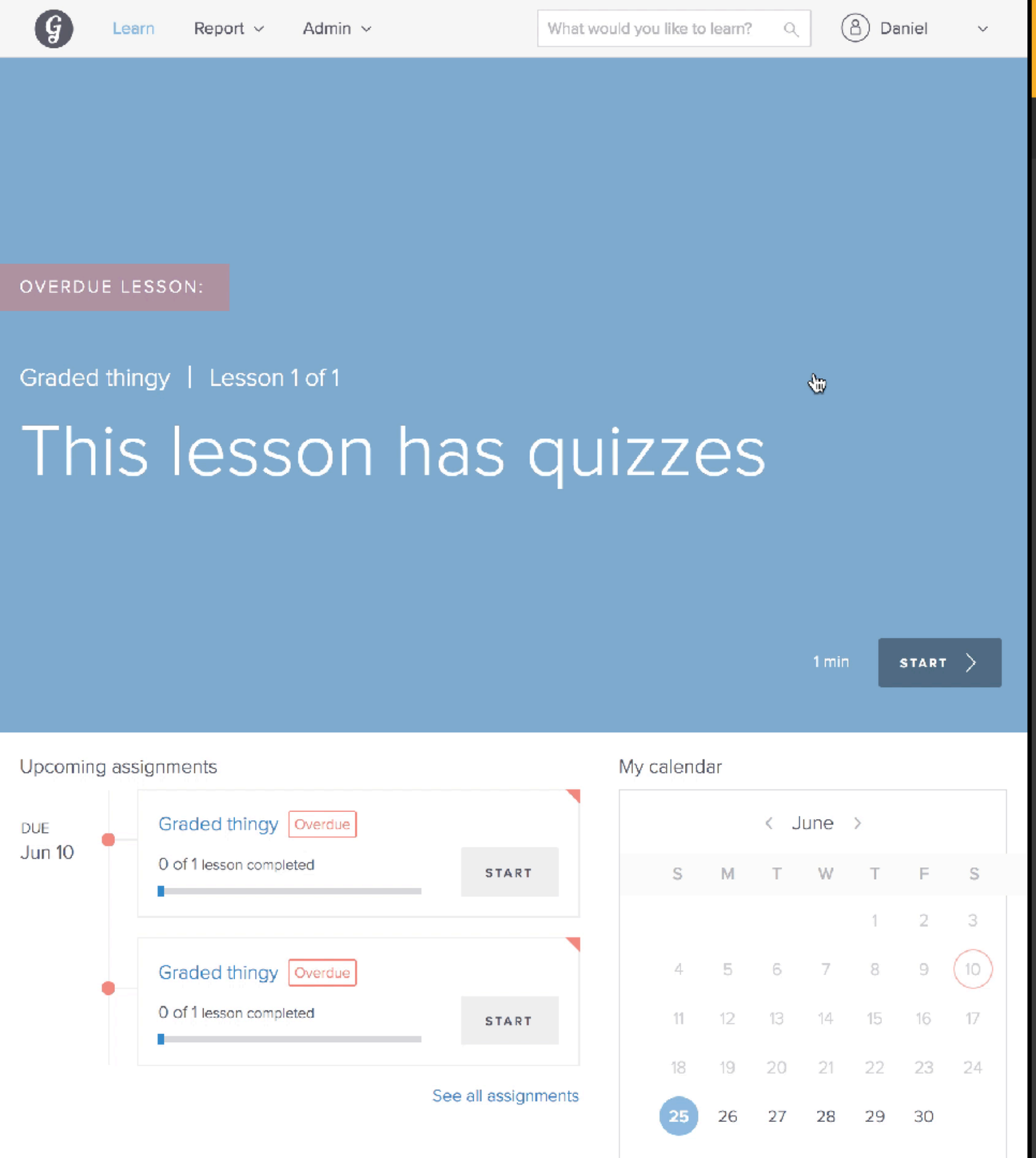
1. Clone this repo
2. Add `127.0.0.1 galaxy.local.net` to your hosts file (follow these instructions if this is new to you)
3. Run `yarn install`
4. Copy `.env.dist` to `.env` and get dev secrets from a member of your grove
5. Run `yarn run client-run`
6. Connect to dev VPN (a member from the infrastructure team can help you set this up)
7. Open your browser to http://galaxy.local.net:1110

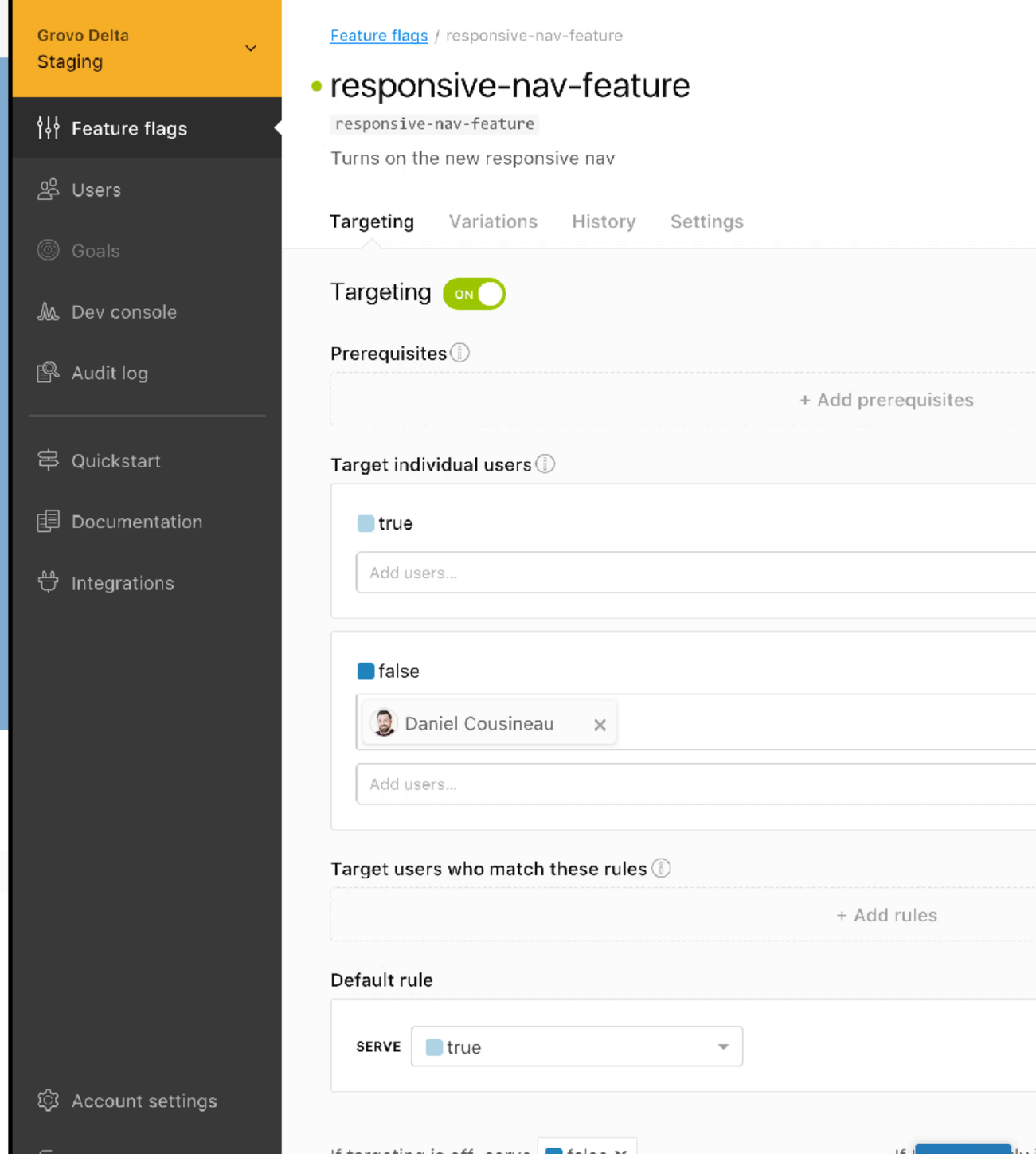
Pre: Clear homebrew & yarn caches

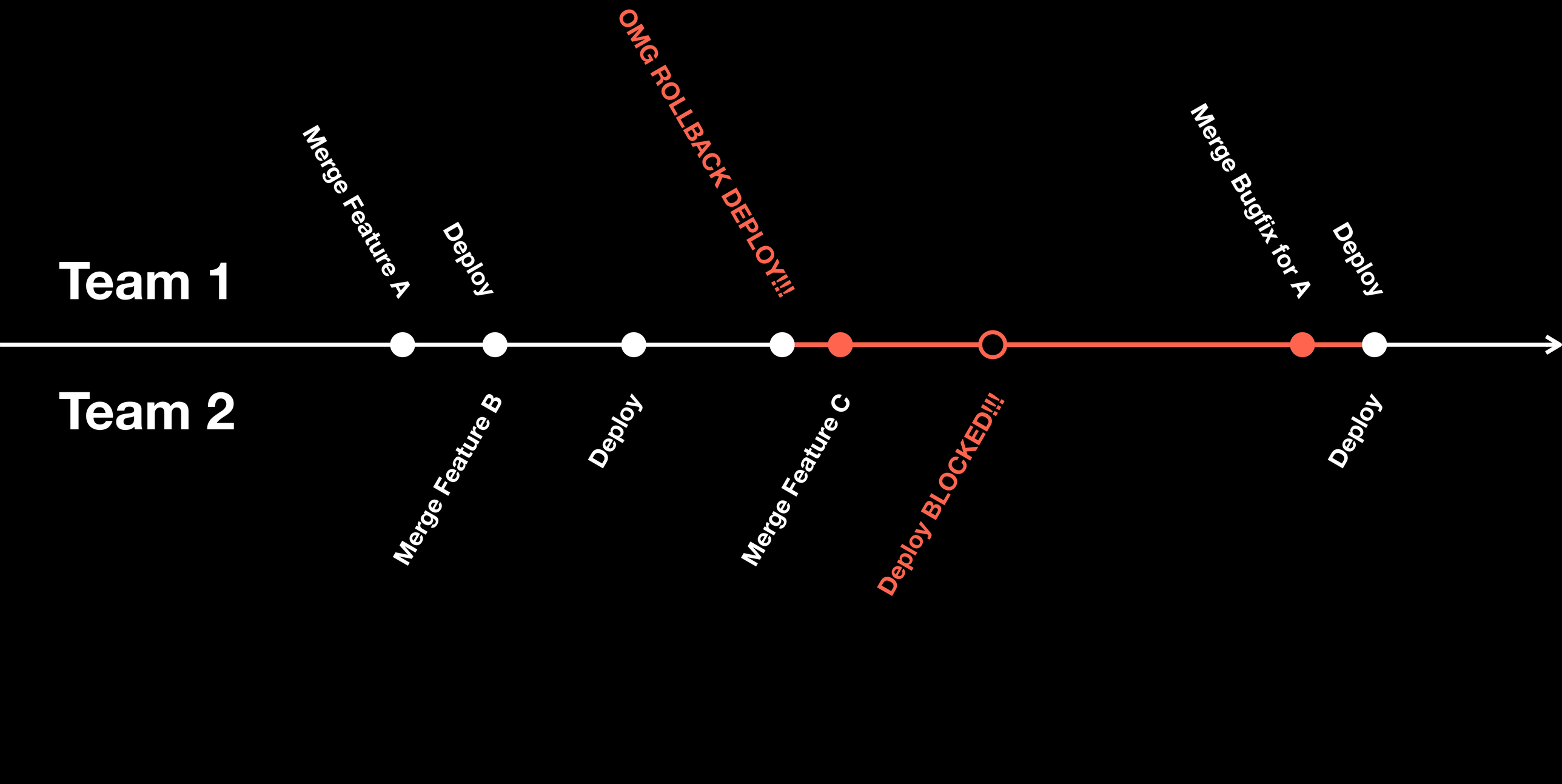
1. **Reinstall node & yarn via `brew`**
2. **Clone repo**
3. **Run `yarn install`**
4. **Run production build**
    1. **Compile & Minify CSS**
    2. **Compile Server via Babel**
    3. **Compile, Minify, & Gzip via Webpack**

190.64s
~3 min

```
<Feature name="new-feature" fallback={<OldFeatureComponent />}>
    <NewFeatureComponent />
</Feature>
```


Learn
Report
Admin
What would you like to learn?
Daniel
OVERDUE LESSON:
Graded thingy | Lesson 1 of 1
This lesson has quizzes
1 min
START
Upcoming assignments
DUE
Jun 10
Graded thingy
Overdue
0 of 1 lesson completed
START
Graded thingy
Overdue
0 of 1 lesson completed
START
See all assignments
My calendar
June
S M T W T F S
1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30


Grovo Delta
Staging
Feature flags
Users
Goals
Dev console
Audit log
Quickstart
Documentation
Integrations
Account settings
Feature flags / responsive-nav-feature
responsive-nav-feature
responsive-nav-feature
Turns on the new responsive nav
Targeting
Variations
History
Settings
Targeting
ON
Prerequisites
+ Add prerequisites
Target individual users
true
Add users...
false
Daniel Cousineau
Add users...
Target users who match these rules
+ Add rules
Default rule
SERVE
true


Team 1
Team 2
Merge Feature A
Deploy
Merge Feature B
Deploy
OMG ROLLBACK DEPLOY!!!
Merge Feature C
Deploy BLOCKED!!!
Merge Bugfix for A
Deploy
Deploy


Team 1
Merge Feature A
Deploy
Rollout Flag A
OMG ROLLBACK FLAG A!!!
Merge Bugfix for A
Deploy
Rollout Flag A
Team 2
Merge Feature B
Deploy
Rollout Flag B
Merge Feature C
Deploy
Rollout Flag C

Can you optimize your directory structure around team responsibilities?

If teams are organized by “product domain”,
Can you organize code around product domain?


Final Thoughts

Strict rules rarely 100% apply to your application.
Remembering the purpose behind the rules is valuable.

Code behavior should be predictable and intuitable.
Be realistic about the problem you're actually solving.

You will not get it perfect the first time.
Optimize your processes for refactoring.


I HAVE NO IDEA WHAT MY GORILLA RECEPTIONIST IS SAYING... BUT I DO KNOW THAT THIS BOY HAS PAUL BUNYAN'S DISEASE, AND IF I DON'T OPERATE RIGHT NOW, HE'LL TRANSFORM INTO A GIANT LUMBERJACK!
OOK! OOK! AH AH!*

Questions?