THUNK MIDDLEWARE

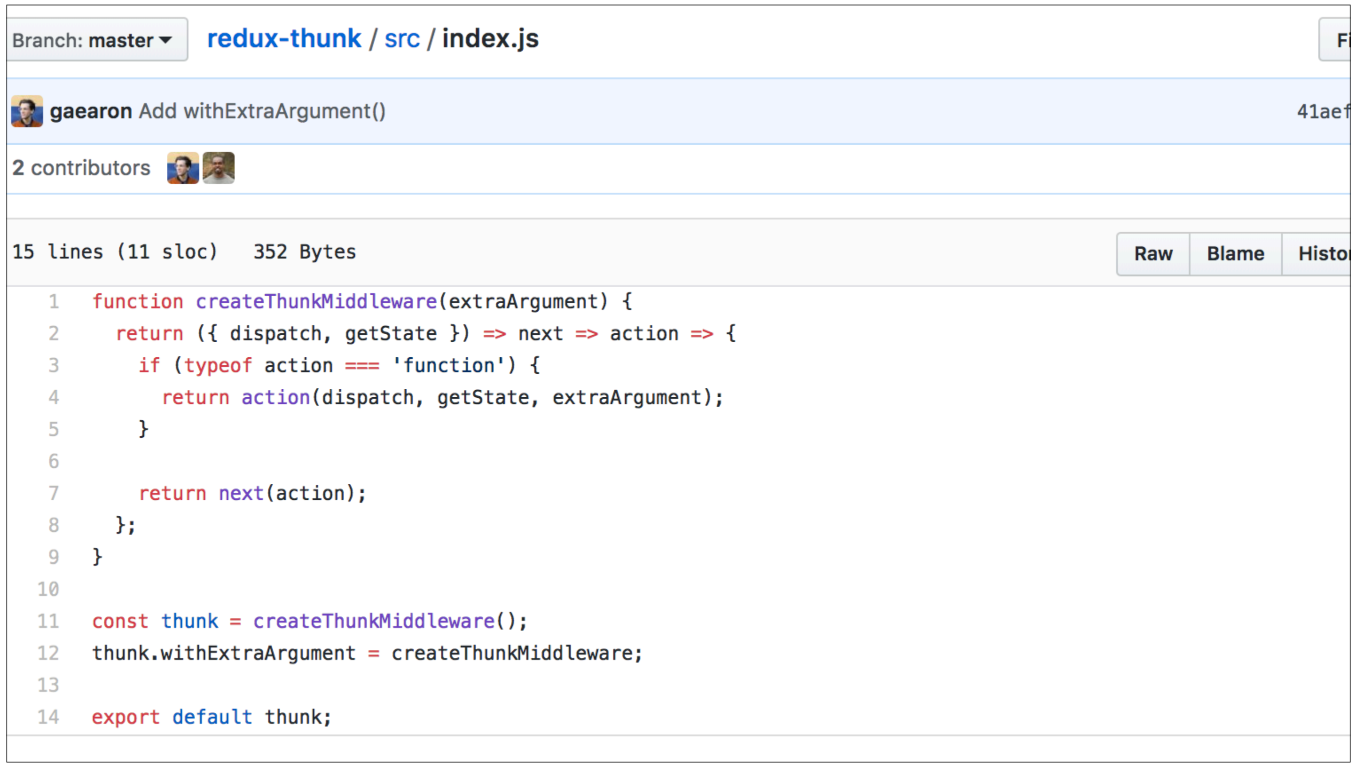
Who'da thunk it?

TRAJECTORY

- What the thunk?
- Why the thunk would I do this?
- When to thunk?

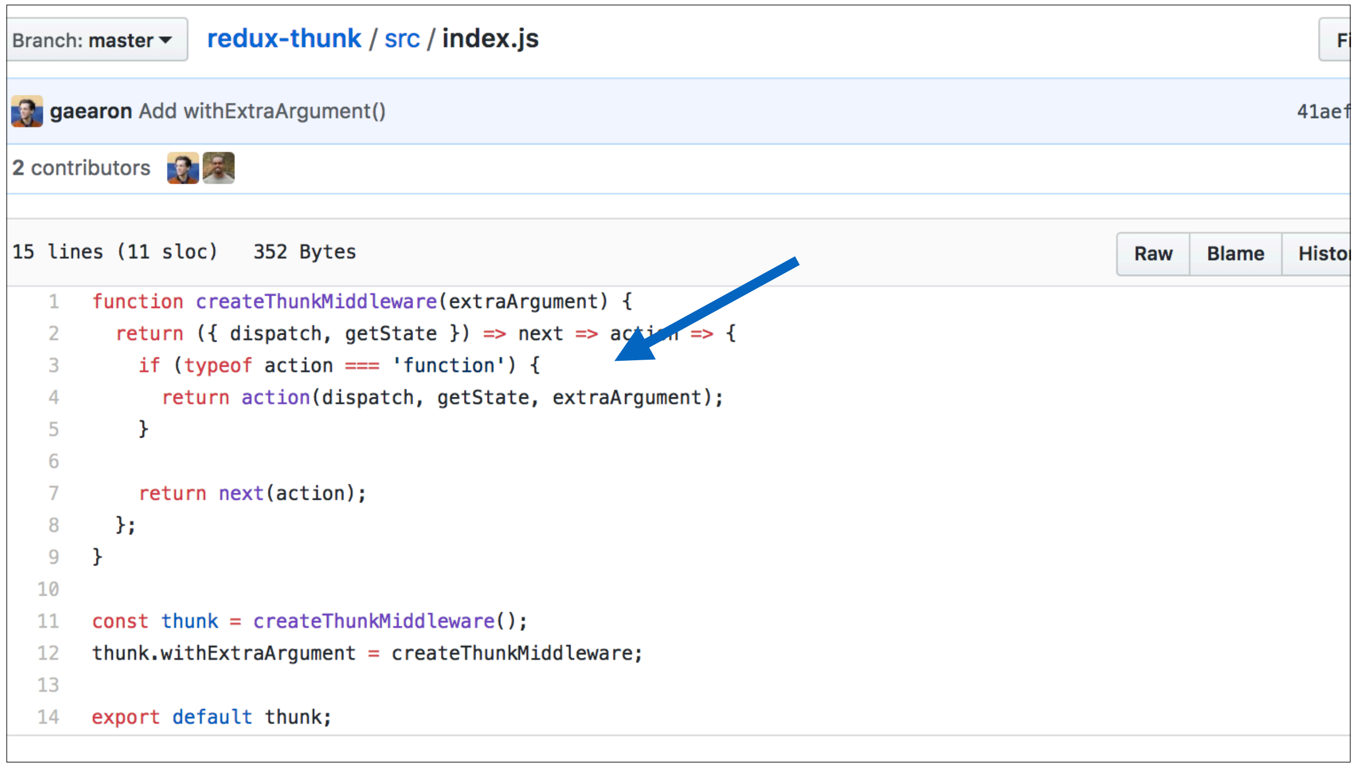
WHAT IS THUNK MIDDLEWARE?

- Checks the incoming action
  - If the action is a regular object, do nothing
  - If the "action" is a function, invoke it, and pass the store's `dispatch` and `getState` methods to it!
  - We call that function a "thunk" (it's just a term borrowed from functional programming and elsewhere)



```
15 lines (11 sloc)    352 Bytes
```

```
function createThunkMiddleware(extraArgument) {
  return ({ dispatch, getState }) => next => action => {
    if (typeof action === 'function') {
      return action(dispatch, getState, extraArgument);
    }

    return next(action);
  };
}

const thunk = createThunkMiddleware();
thunk.withExtraArgument = createThunkMiddleware;

export default thunk;
```



redux-thunk / src / index.js

gaearon Add withExtraArgument()

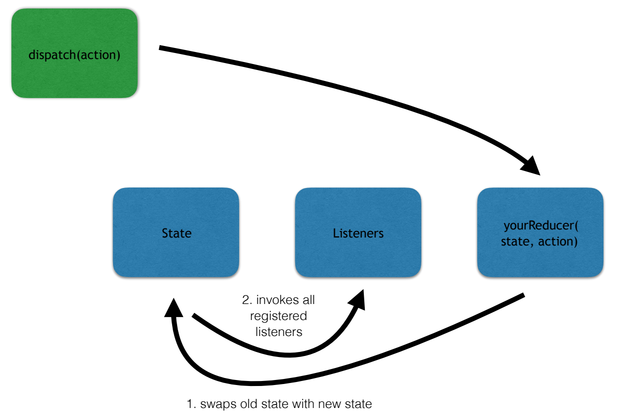
41aef

2 contributors

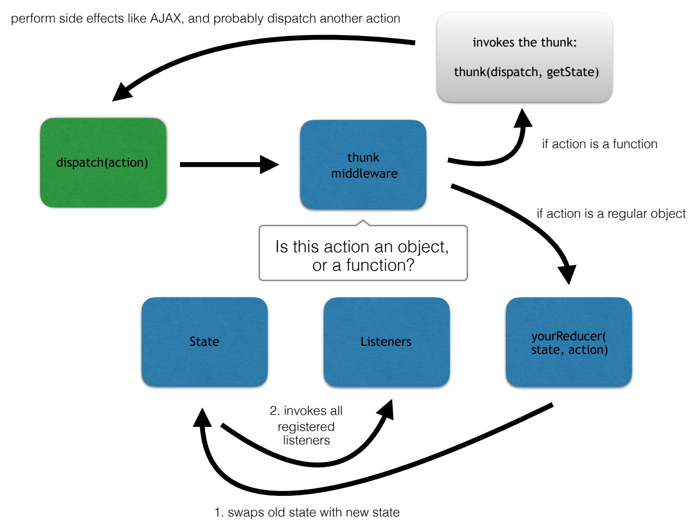
15 lines (11 sloc)   352 Bytes

```
function createThunkMiddleware(extraArgument) {
  return ({ dispatch, getState }) => next => action => {
    if (typeof action === 'function') {
      return action(dispatch, getState, extraArgument);
    }

    return next(action);
  };
}

const thunk = createThunkMiddleware();
thunk.withExtraArgument = createThunkMiddleware;

export default thunk;
```

WITHOUT THUNK MIDDLEWARE


dispatch(action)
State
Listeners
yourReducer(state, action)
2. invokes all registered listeners
1. swaps old state with new state

WITH THUNK MIDDLEWARE


perform side effects like AJAX, and probably dispatch another action
invokes the thunk:
thunk(dispatch, getState)
dispatch(action)
thunk middleware
if action is a function
if action is a regular object
Is this action an object, or a function?
State
Listeners
yourReducer(state, action)
2. invokes all registered listeners
1. swaps old state with new state

WHY THUNK?

- **Helps you stay DRY**
  - Several different components might want to make the same API requests.
- **Removes “responsibility” for asynchronous code/side effects from components**
  - Components don’t need to know whether an action is async or not - they just dispatch!

TERMINOLOGY

- Thunk creator: an action creator that returns a thunk; also known as “async action creators”
- Thunk: the function returned from a thunk creator; it accepts `dispatch` and `getState` as arguments

```
const GET_PUGS = 'GET_PUGS'


const gotPugs = (pugs) => {

  return {
    type: GET_PUGS,
    pugs
  }
}
```

```
// the "action type”
const GET_PUGS = 'GET_PUGS'


const gotPugs = (pugs) => {

  return {
    type: GET_PUGS,
    pugs
  }
}
```

```
// the "action type”
const GET_PUGS = 'GET_PUGS'

// the "action creator"
const gotPugs = (pugs) => {

  return {
    type: GET_PUGS,
    pugs
  }
}
```

```
// the "action type”
const GET_PUGS = 'GET_PUGS'

// the "action creator"
const gotPugs = (pugs) => {
  // the "action"
  return {
    type: GET_PUGS,
    pugs
  }
}
```

```
// the "action type”
const GET_PUGS = 'GET_PUGS'

// the "action creator"
const gotPugs = (pugs) => {
  // the "action"
  return {
    type: GET_PUGS,
    pugs
  }
}
```

```
const getPugs = () => {




}
```

```
// the "action type”
const GET_PUGS = 'GET_PUGS'

// the "action creator"
const gotPugs = (pugs) => {
  // the "action"
  return {
    type: GET_PUGS,
    pugs
  }
}
```

```
const getPugs = () => {

  return async (dispatch, getState) => {


  }
}
```

```
// the "action type”
const GET_PUGS = 'GET_PUGS'

// the "action creator"
const gotPugs = (pugs) => {
  // the "action"
  return {
    type: GET_PUGS,
    pugs
  }
}
```

```
const getPugs = () => {

  return async (dispatch, getState) => {
    const {data} = await axios.get('/api/pugs')
    dispatch(gotPugs(data))
  }
}
```

```
// the "action type”
const GET_PUGS = 'GET_PUGS'

// the "action creator"
const gotPugs = (pugs) => {
  // the "action"
  return {
    type: GET_PUGS,
    pugs
  }
}
```

```
// the "thunk creator"
const getPugs = () => {

  return async (dispatch, getState) => {
    const {data} = await axios.get('/api/pugs')
    dispatch(gotPugs(data))
  }
}
```

```
// the "action type”
const GET_PUGS = 'GET_PUGS'

// the "action creator"
const gotPugs = (pugs) => {
  // the "action"
  return {
    type: GET_PUGS,
    pugs
  }
}
```

```
// the "thunk creator"
const getPugs = () => {
  // the "thunk"
  return async (dispatch, getState) => {
    const {data} = await axios.get('/api/pugs')
    dispatch(gotPugs(data))
  }
}
```

```
import React from 'react'
import {connect} from 'react-redux'
import {getPugs} from '../store'
import React from 'react'
import {connect} from 'react-redux'
import {getPugs} from '../store'

const AllPugs = connect(










)(




 )
```

```
import React from 'react'
import {connect} from 'react-redux'
import {getPugs} from '../store'

const AllPugs = connect(









)(class extends React.Component {




})
```

```
import React from 'react'
import {connect} from 'react-redux'
import {getPugs} from '../store'

const AllPugs = connect(
  // mapStateToProps
  (state) => {
    return {
      pugs: state.pugs
    }
  },




)(class extends React.Component {




})
```

```
import React from 'react'
import {connect} from 'react-redux'
import {getPugs} from '../store'

const AllPugs = connect(
  // mapStateToProps
  (state) => {
    return {
      pugs: state.pugs
    }
  },
  // mapDispatchToProps
  (dispatch) => ({
    getPugs: () => dispatch(getPugs())
  })
)(class extends React.Component {




})
```

```
import React from 'react'
import {connect} from 'react-redux'
import {getPugs} from '../store'

const AllPugs = connect(
  // mapStateToProps
  (state) => {
    return {
      pugs: state.pugs
    }
  },
  // mapDispatchToProps
  (dispatch) => ({
    getPugs: () => dispatch(getPugs())
  })
)(class extends React.Component {
  componentDidMount () {
    this.props.getPugs()
  }
  // etc....
})
```

MUST I ALWAYS THUNK?

- No, not always
- Thunks, like Redux itself, are tools to help organize big applications
  - Sitting down to write your company's big analytics dashboard app? Thunk it up!
  - Writing something fun for your personal site? Not the time to be thunky. Not even the time for Redux!