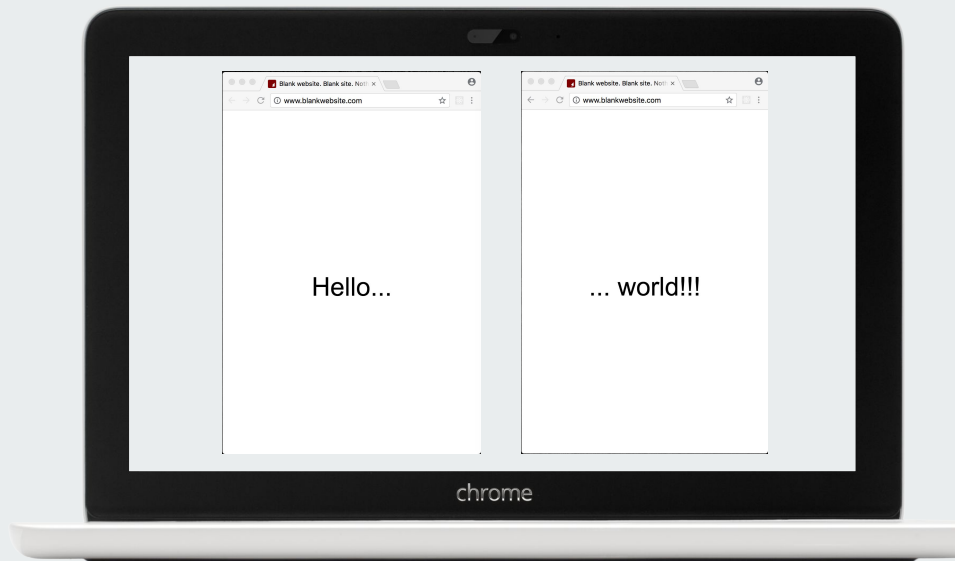




If these tabs could talk?

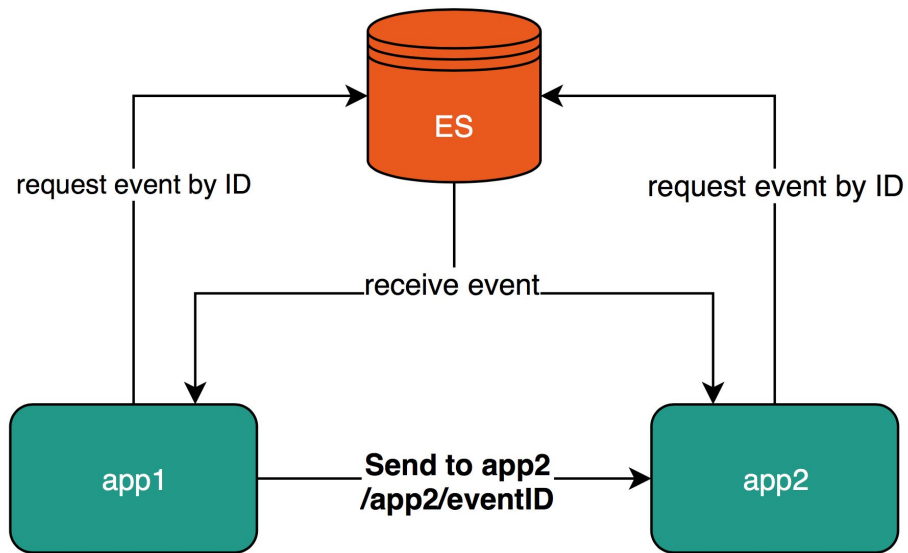
Presented by: Zachary Sierakowski

06/21/2018

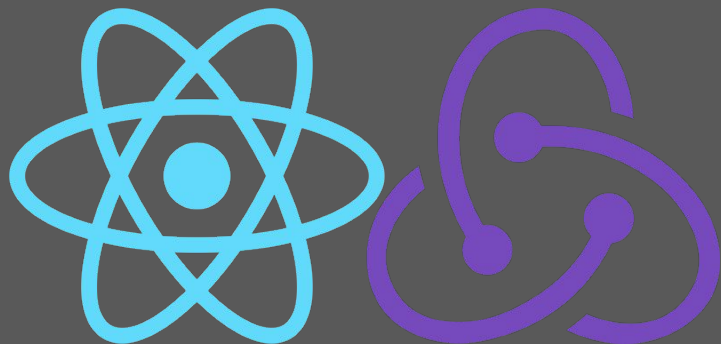


Motivation

- Deep Linking
- Redux and sharing redux state
- No Local Storage



Outline



Redux

- Relation to React
- Actions/Reducers

Redux Middleware

- What and why?
- Examples

Tab Communication

- Implementations
- Web Workers

Redux & Web Workers

- Motivation
- Demo



Redux



Relation to React

Used by React applications to manage state with a single store

- State is read-only... never mutate it
- To update the state, emit objects (**actions**) describing the change
- State is transformed as a result of these actions through pure functions (**reducers**) that return new state

Action ('type' is required)

```
{  
  type: ADD_TODO,  
  text: 'Im a redux action'  
}
```

Reducer

```
function todoApp(state = initialState, action) {  
  switch (action.type) {  
    case ADD_TODO:  
      return Object.assign({}, state, {  
        text: action.text  
      })  
    default:  
      return state  
  }  
}
```

Redux Middleware



What is Redux Middleware?

- Extension point between dispatching an action, and the moment it reaches the reducer
- Composable Chain



Why use Redux Middleware?

- Side-effects to any/all actions -- in a singular location
- Triggering an actions from an action (careful here)
- Example use-cases:
 - logging
 - crash reporting
 - async APIs
 - routing



Examples

creating middleware

Middleware to log every
action and the resulting
state after the action.

```
const logger = store => next => action => {  
  console.log('dispatching', action);  
  let result = next(action);  
  console.log('next state', store.getState());  
  return result;  
}
```



Examples

creating middleware

Middleware to send
metrics to a DB based on
different action types.

```
const metricMiddleware = store => next => action => {  
  if(action.type.includes('CLICK')) {  
    client.click(action);  
  }  
  if(action.type.includes('ERROR')) {  
    client.reportError(action);  
  }  
  return next(action);  
}
```



Examples

applying middleware

applyMiddleware() tells
createStore() how to
handle middleware

```
import { createStore, applyMiddleware } from 'redux';  
const store = createStore(  
  reducers,  
  applyMiddleware(logger, metricMiddleware)  
);
```



Chaining Middleware

The middleware pipeline exactly matches the order that you passed to `applyMiddleware()`. So, in this example:

- calling `store.dispatch()` passes the action to `logger`
- when `logger` calls `next(action)`, it goes to `metricMiddleware`
- when `metricMiddleware` calls `next(action)`, it goes to the store and the reducer logic is executed

Tab Communication



Implementations

- Local Storage Events
 - Most browser support
- Broadcast Channel API
 - Easy to use and no worker.js file needed
 - Little browser support -- see caniuse.com
- Shared Web Workers API
 - Reasonable browser support -- see caniuse.com



Shared Web Workers

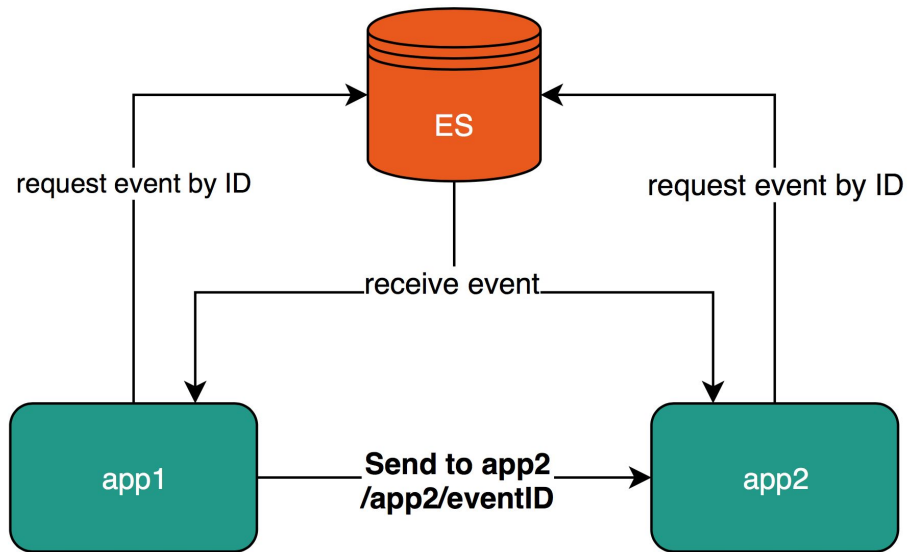
- Web Workers are a simple means for web content to run scripts in background threads
- Shared Web Workers have essentially the same API as Web Workers

```
var myWorker = new SharedWorker('/path/to/worker.js');  
  
myWorker.port.start();  
  
myWorker.port.postMessage("hello world");  
  
myWorker.port.onmessage = function(e) {  
  console.log('Message received from worker -- ', e.data);  
}
```

Redux & Web Workers

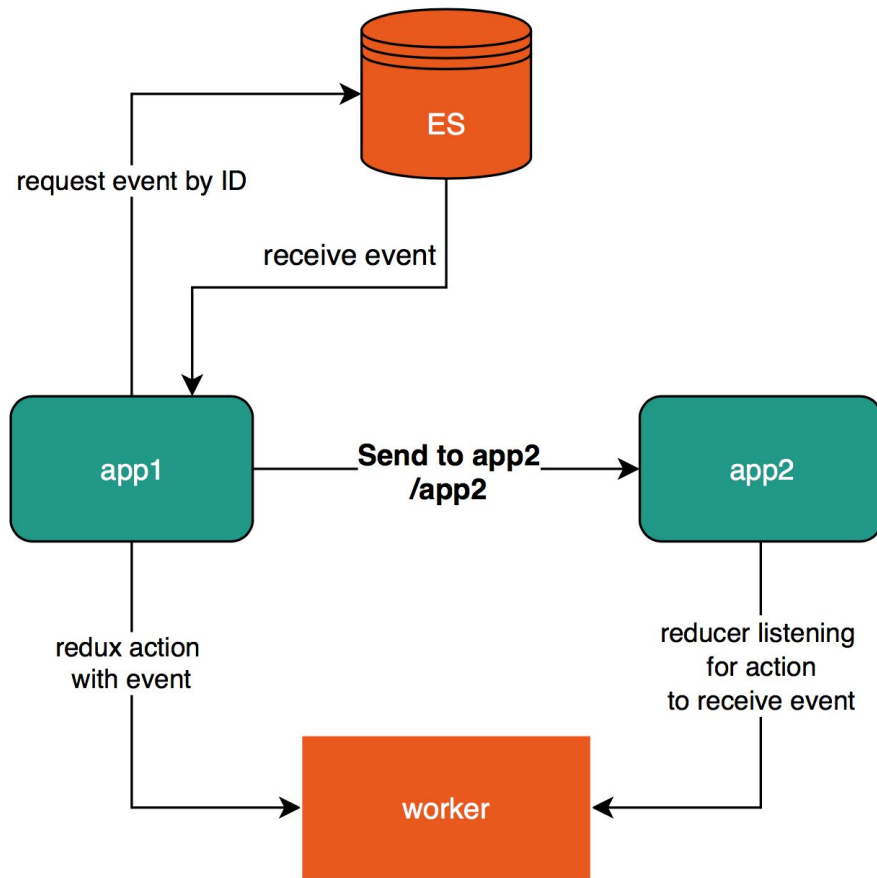
Motivation

- Deep Linking
- Redux and sharing redux state
- No Local Storage



Motivation

- Deep Linking
- Redux and sharing redux state
- No Local Storage





Demo



What next?

- Find your own use cases for Redux Middleware and create your own!
- Web Workers can be used in other ways as well
- If you're interested in other ways we have used Redux Middleware or Web Workers in our applications, feel free to ask! (yes, there is more)
- You can find the demo source and download/view this presentation [here](#)

Questions?



References

[redux](#)

[redux-logger](#)

[web workers](#)

[caniuse.com](#)