

MANAGE STATE

# Angular Redux

Christina Kayastha  
Senior Software Engineer  
Vistaprint, Cimpress  
@christikaes

# HELLO WORLD!

My name is Christina (:

Christina Kayastha

Senior Software Engineer

Vistaprint, Boston, MA

@christikaes

I'm all about:

- Icecream
- Community Events
- Bleeding Edge Technology



1

# BUILD FOR MOBILE Progressive Web App



2

## MANAGE STATE Redux



3

## USER INTERFACE Material



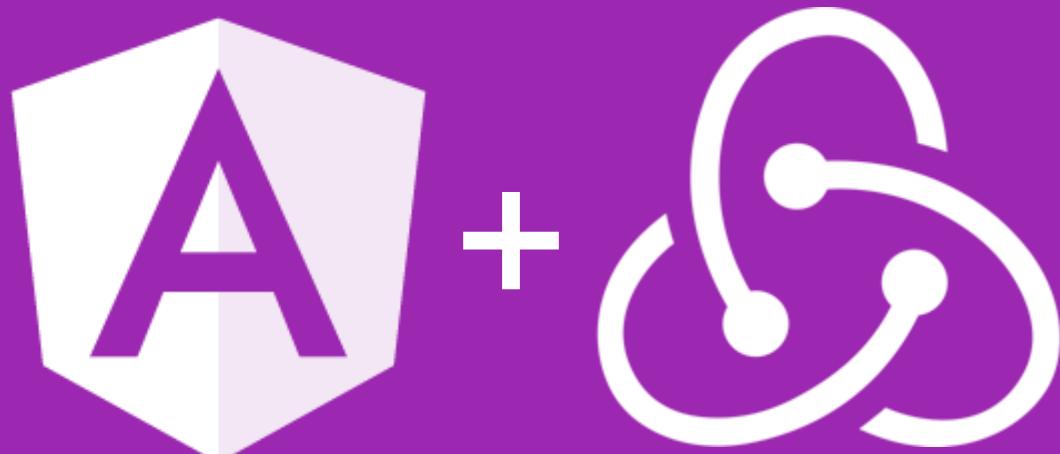
4

## AUTH & DATABASE Firebase



5

## ANGULAR APP SPEED RUN!! Everything



MANAGE STATE

# Angular Redux

Christina Kayastha  
Senior Software Engineer  
Vistaprint, Cimpress  
@christikaes



# Angular Redux

**WHAT is it ?**

**HOW can I use it ?**

**WHY should I care ?**



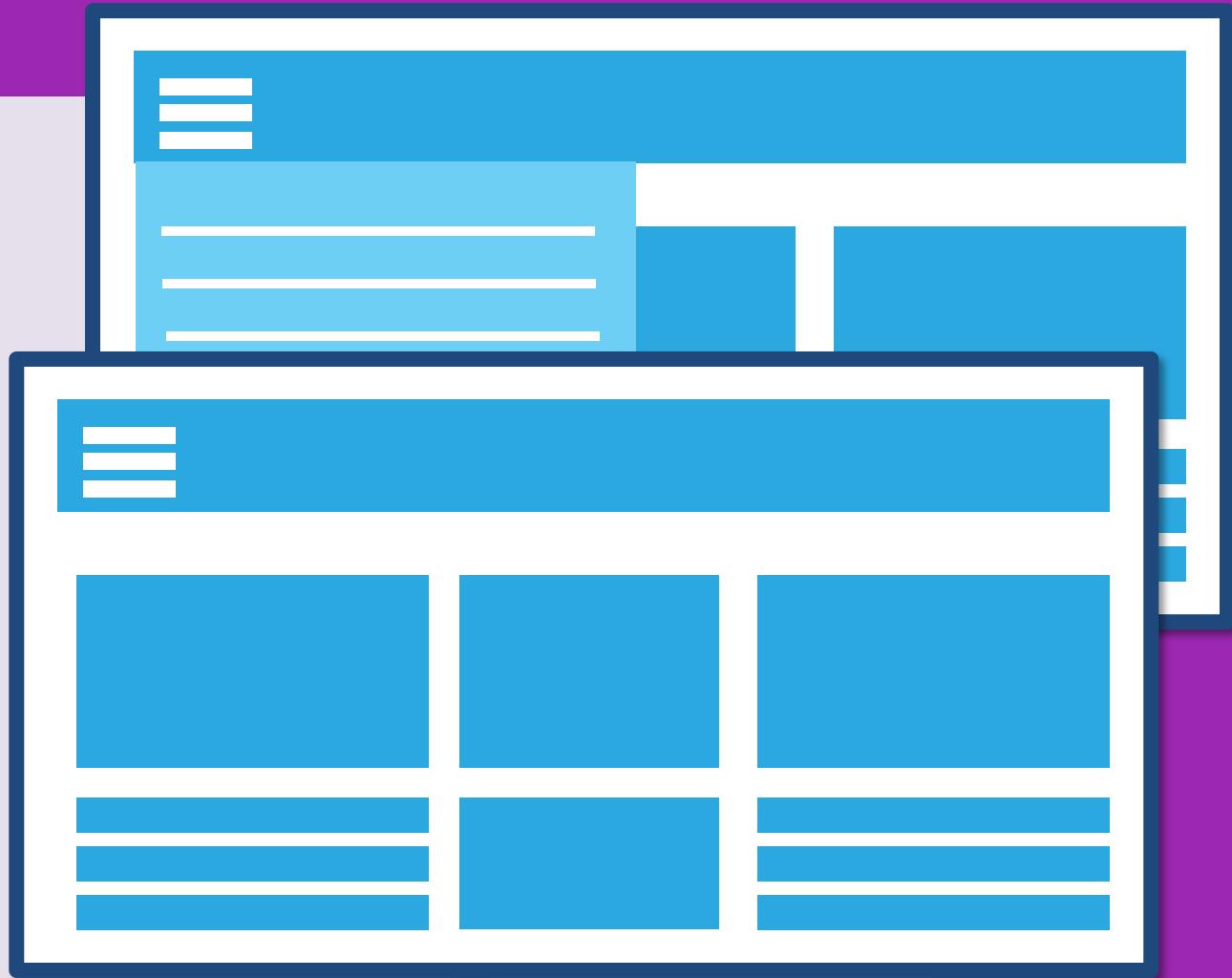
What is  
STATE ?



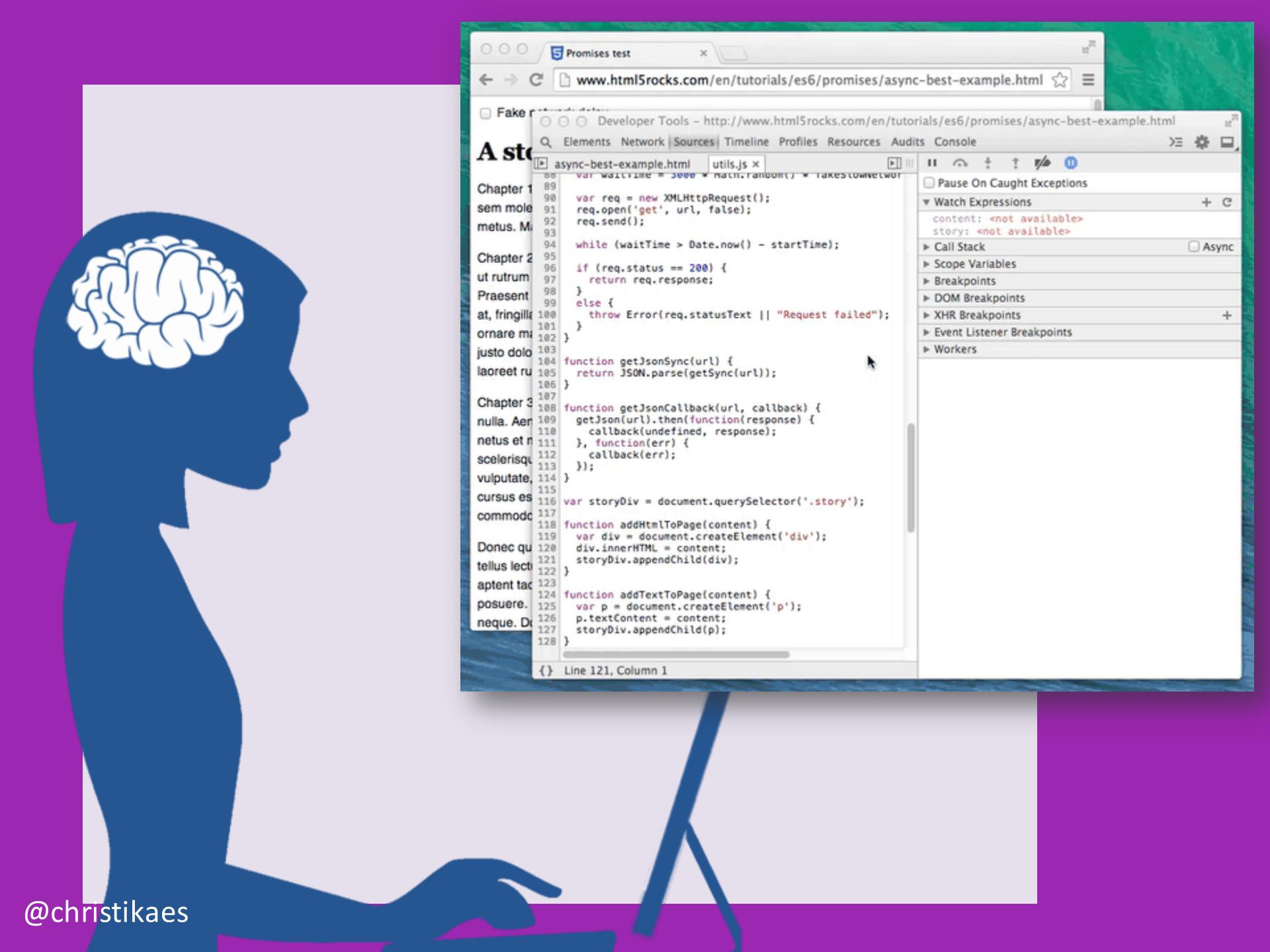
State is what an application knows about the user, interactions, and other pieces of global information



Why should  
I care about  
**MANAGING STATE ?**

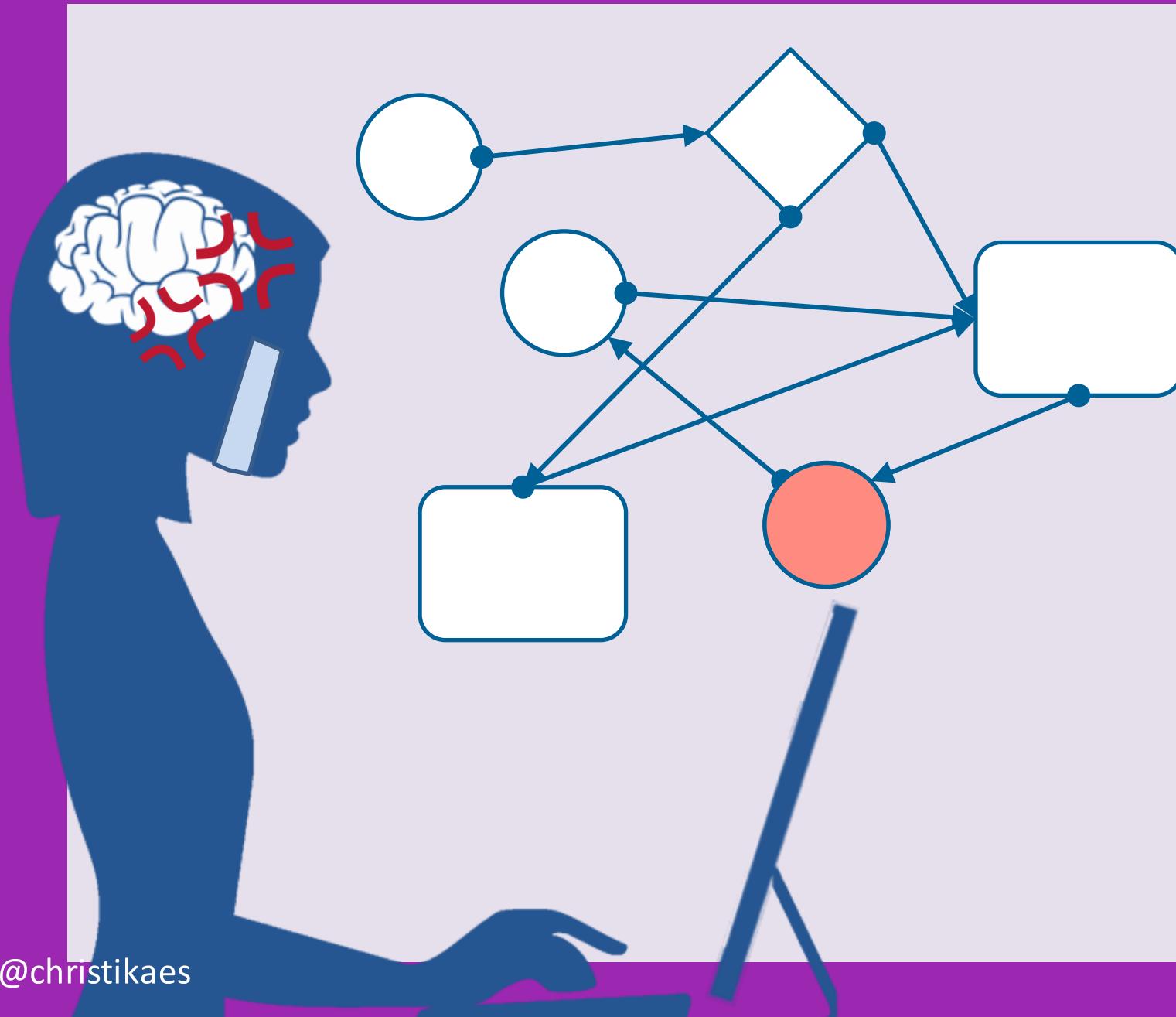


@christikaes

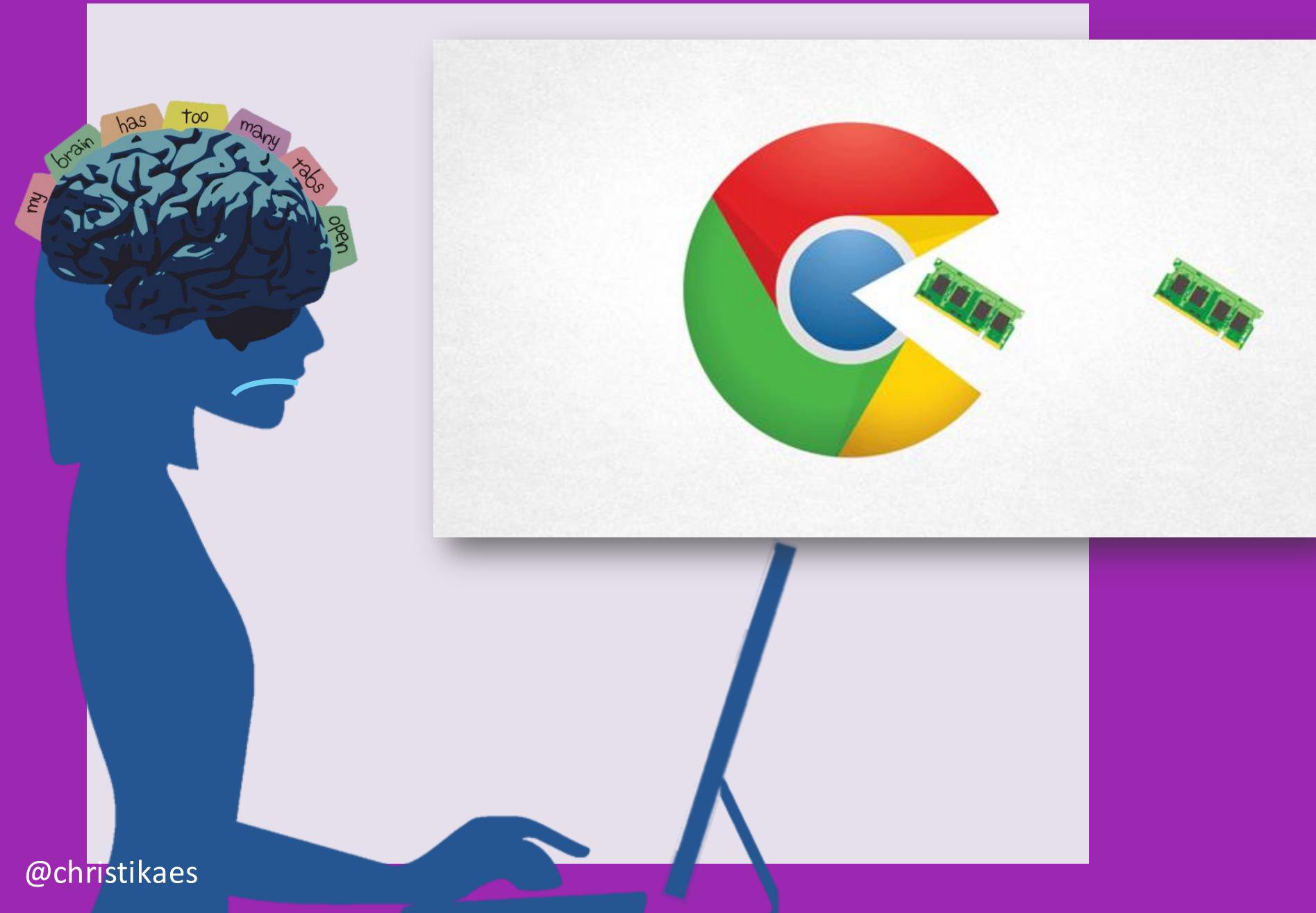


A screenshot of a web browser window titled "Promises test" showing a developer tools interface. The main content area displays a portion of a JavaScript file named "utils.js" with line numbers 88 to 128. The code uses XMLHttpRequest to make asynchronous requests and handle their responses. The browser's address bar shows the URL "www.html5rocks.com/en/tutorials/es6/promises/async-best-example.html". The developer tools sidebar on the right contains sections for "Watch Expressions", "Call Stack", "Scope Variables", "Breakpoints", "DOM Breakpoints", "XHR Breakpoints", "Event Listener Breakpoints", and "Workers".

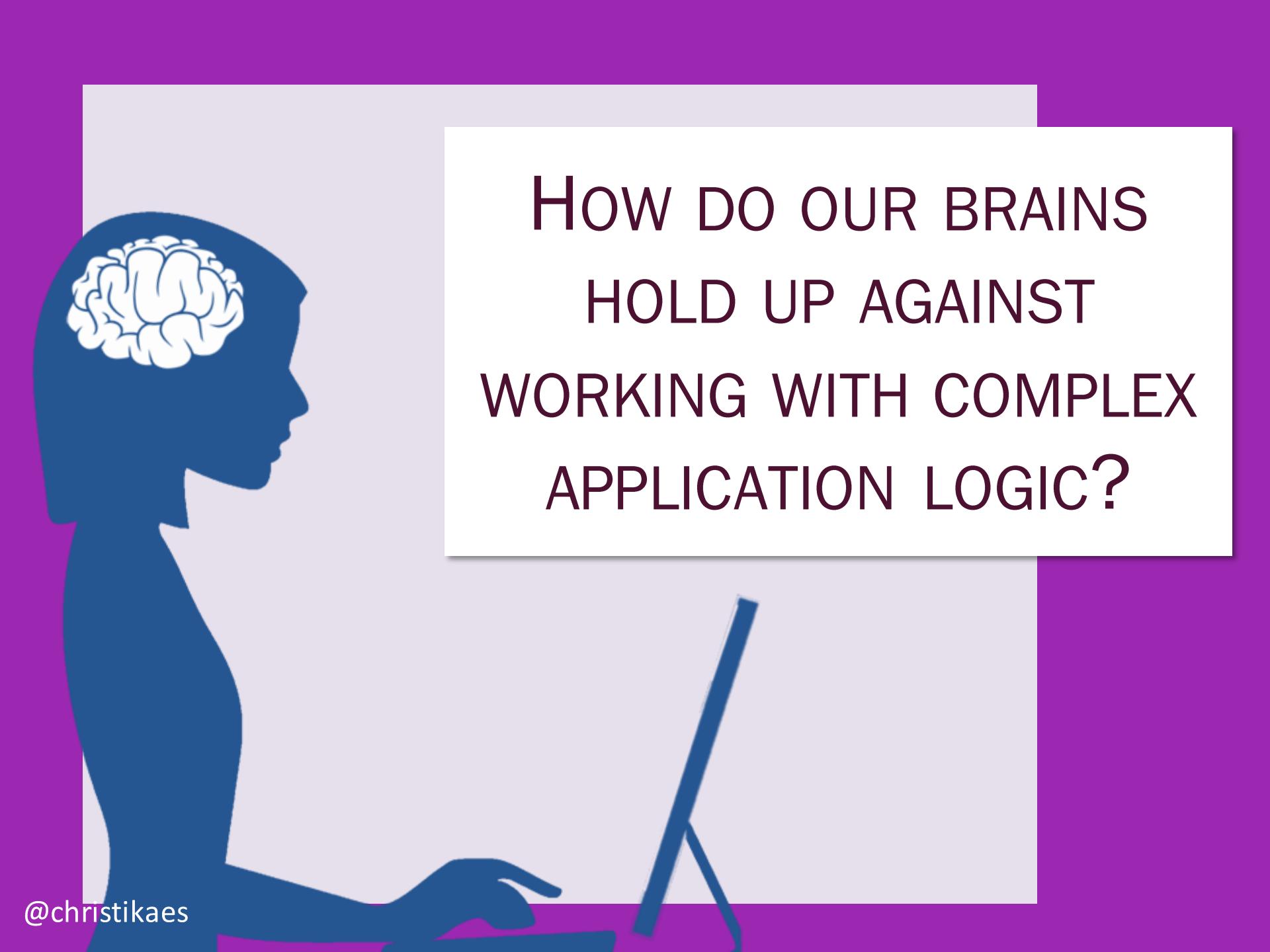
```
88 var waitTime = 3000 * Math.random() * randomWaitFactor;
89
90 sem mole
91 req.open('get', url, false);
92 req.send();
93
94 while (waitTime > Date.now() - startTime);
95
96 if (req.status == 200) {
97   return req.response;
98 } else {
99   throw Error(req.statusText || "Request failed");
100 }
101
102
103 function getJsonSync(url) {
104   return JSON.parse(getSync(url));
105 }
106
107
108 function getJsonCallback(url, callback) {
109   getJson(url).then(function(response) {
110     callback(undefined, response);
111   }, function(err) {
112     callback(err);
113   });
114 }
115
116 var storyDiv = document.querySelector('.story');
117
118 function addHtmlToPage(content) {
119   var div = document.createElement('div');
120   div.innerHTML = content;
121   storyDiv.appendChild(div);
122 }
123
124 function addTextToPage(content) {
125   var p = document.createElement('p');
126   p.textContent = content;
127   storyDiv.appendChild(p);
128 }
```



@christikaes



@christikaes

A large silhouette of a person's head and shoulders is positioned on the left side of the slide. The head is turned slightly to the right, showing a white brain inside. The person is sitting at a desk, with their hands visible on a keyboard. A monitor is on the desk, displaying a single blue line graph.

# How do our brains hold up against working with complex application logic?

# STAND BACK



# I'M GOING TO TRY SCIENCE

# WHAT PARTS OF THE BRAIN DO PROGRAMMERS ACTIVATE?

## Understanding Understanding Source Code with Functional Magnetic Resonance Imaging

Janet Siegmund<sup>π\*</sup>, Christian Kästner<sup>ω</sup>, Sven Apel<sup>π</sup>, Chris Parnin<sup>β</sup>, Anja Bethmann<sup>θ</sup>, Thomas Leich<sup>δ</sup>, Gunter Saake<sup>σ</sup>, and André Brechmann<sup>θ</sup>

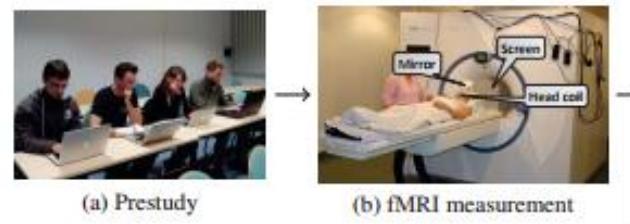
<sup>\*</sup>University of Passau, Germany    <sup>ω</sup>Carnegie Mellon University, USA

<sup>β</sup>Georgia Institute of Technology, USA    <sup>θ</sup>Leibniz Inst. for Neurobiology Magdeburg, Germany

<sup>δ</sup>Metop Research Institute, Magdeburg, Germany    <sup>σ</sup>University of Magdeburg, Germany

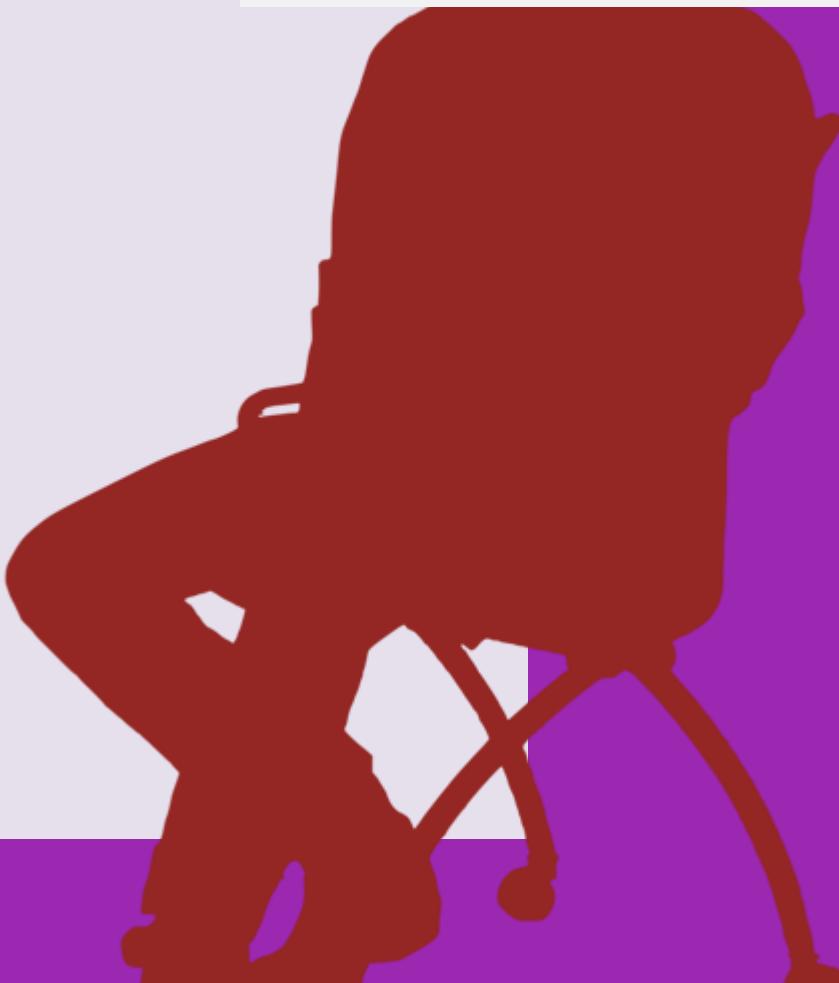
### ABSTRACT

Program comprehension is an important cognitive process that inherently eludes direct measurement. Thus, researchers are struggling with providing suitable programming languages, tools, or coding conventions to support developers in their everyday work. In this paper, we explore whether *functional magnetic resonance imaging (fMRI)*, which is well established in cognitive neuroscience, is feasible to soundly measure program comprehension. In a controlled experiment, we observed 17 participants inside an fMRI scanner while they were comprehending short source-code snippets, which we contrasted with locating syntax errors. We found a clear, distinct activation pattern of five brain regions, which are related to working memory, attention, and language processing—all processes that fit well to our understanding of program comprehension. Our results encourage us and, hopefully, other researchers to use fMRI in future studies to measure program comprehension and, in particular, to compare it to other cognitive measures.

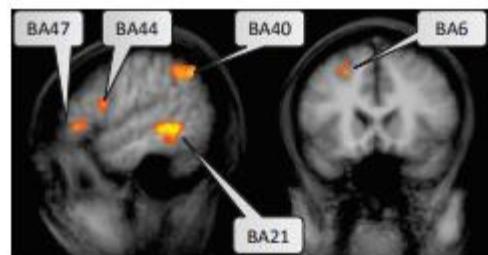
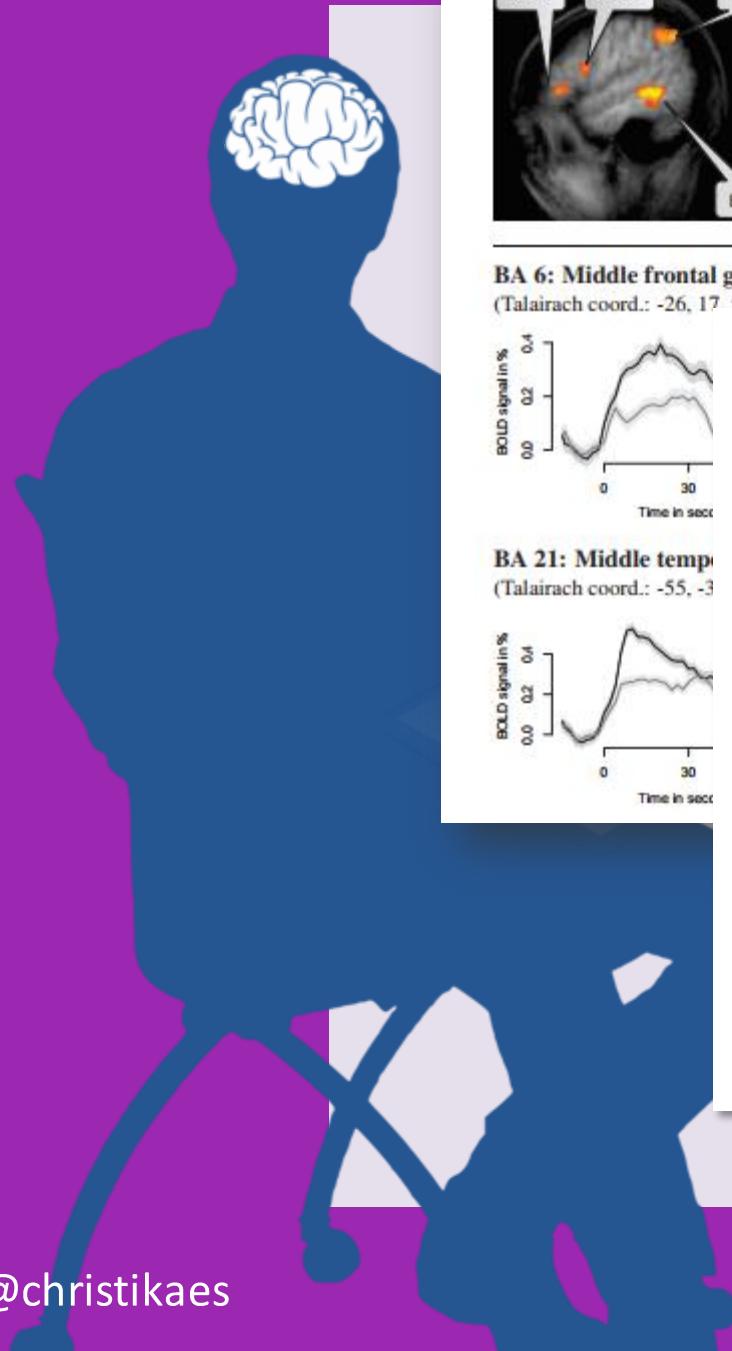




@christikaes

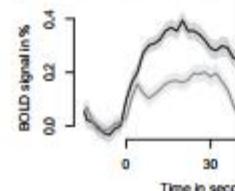


@christikaes



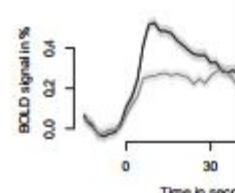
**BA 6: Middle frontal gyrus**

(Talairach coord.: -26, 17, 52; cluster size: 1270)



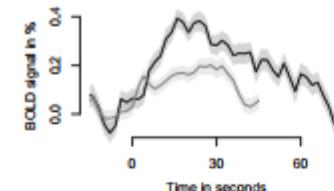
**BA 21: Middle tempo**

(Talairach coord.: -55, -3)



**BA 40: Inferior parietal lobule**

(Talairach coord.: -51, -49, 41; cluster size: 3368)



Working memory  
Verbal/numeric  
Problem solving

**BA 44: Inferior frontal gyrus**

(Talairach coord.: -50, 11, 16; cluster size: 698)

# ATTENTION WORKING MEMORY SEMANTIC MEMORY LANGUAGE

# WHEN DO WE USE THE MOST ATTENTION/WORKING MEMORY?

CHI 2004 | Late Breaking Results Paper

24-29 April | Vienna, Austria

2011 19th IEEE International Conference on Program Comprehension

CHI 2007 Proceedings • Tasks

April 28-May 3, 2007 • San Jose, CA, USA

## Understanding and Developing Models for Detecting and Differentiating Breakpoints during Interactive Tasks

Shamsi T. Iqbal and Brian P. Bailey

Department of Computer Science

University of Illinois

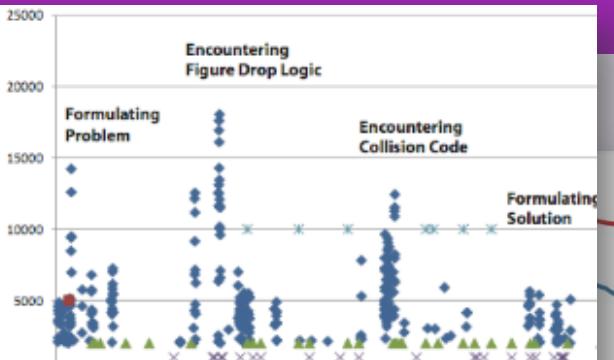
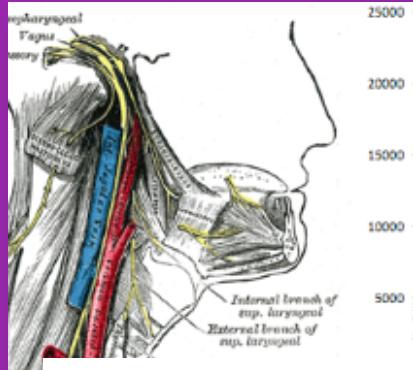
Urbana, IL 61801 USA

{siqbal, bpbailey}@cs.uiuc.edu

### ABSTRACT

The ability to detect and differentiate breakpoints during task execution is critical for enabling defer-to-breakpoint policies within interruption management. In this work, we examine the feasibility of building statistical models that can detect and differentiate three granularities (types) of perceptually meaningful breakpoints during task execution, without having to recognize the underlying tasks. We collected ecological samples of task execution data, and

One common method for detecting breakpoints is to match users' ongoing interaction to specifications of tasks defined a priori [4]. Although this allows breakpoints to be easily detected within tasks that are fairly prescribed, it is much more difficult to leverage these types of static specifications to detect breakpoints within tasks that have highly variable interaction, i.e., *free-form* tasks, yet these are by far the most common type of computing task performed [8]. This



# COMPREHENDING DATA FLOW AND CONTROL FLOW IN CODE



# COMPLEX APPLICATION LOGIC IS TAXING ON OUR HUMAN BRAINS!



WE LIVE IN A WORLD OF  
INCREASINGLY COMPLEX  
FRONTEND APPLICATIONS



# HOW CAN WE MANAGE INCREASINGLY COMPLEX FRONTEND APPLICATIONS



# **REDUX TO MANAGE INCREASINGLY COMPLEX FRONTEND APPLICATIONS**





Managing State is makes  
our app much easier to  
Learn and Debug!



What does  
**REDUX**  
mean anyway?

# Redux?

Brought back, revived

## ReactFlux



Redux is a predictable  
state container for web  
applications



# HOW do I use Redux?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

`github.com/christinakayastha  
/StudioReduxExample`

**HELLO WORLD<sup>+</sup>**  
**LET'S GET STARTED!**

**REDUX IS A  
PREDICTABLE  
STATE CONTAINER  
FOR JAVASCRIPT APPS**

`(state, action) => newState`

ACTION

REDUCER

STORE



`(state, action) => newState`

ACTION

REDUCER

STORE

ALL UPDATES TO THE APPLICATION  
STATE ARE CAUSED BY **ACTIONS**

# JS OBJECT THAT DESCRIBES WHAT HAPPENED?

```
{  
  type: "ADD_TODO",  
  payload: {  
    text: "Learn Redux!"  
  }  
}
```

# BEST PRACTICE: USE ACTION CREATORS TO GENERATE ACTIONS

```
addTodoAction = () =>  
  return {  
    type: "ADD_TODO",  
    payload: {  
      text: "Learn Redux!"  
    }  
  }
```

`(state, action) => newState`

ACTION

REDUCER

STORE

ALL UPDATES TO THE APPLICATION  
STATE ARE CAUSED BY **ACTIONS**

`(state, action) => newState`

ACTION

REDUCER

STORE

THE **REDUCER** TAKES THE  
CURRENT STATE AND AN ACTION,  
AND RETURNS A NEW STATE

# DESIGN THE STATE SHAPE

```
state = [  
  {  
    text: "Learn Redux"  
  }  
  ...  
]
```

YOUR ENTIRE STATE IN 1 JS BLOB

# A FUNCTION THAT DESCRIBES: WHAT'S THE NEW STATE?

```
reducer = (state, action) => {  
  switch(action.type) {  
    case "ADD_TODO":  
      return [  
        ...state,  
        action.payload  
      ];  
    default: return state;  
  }  
}
```

# BEST PRACTICE: USE COMBINEREDUCERS

```
reducer = combineReducers({  
  reducer1,  
  reducer2,  
  ...  
})
```

`(state, action) => newState`

ACTION

REDUCER

STORE

THE **REDUCER** TAKES THE  
CURRENT STATE AND AN ACTION,  
AND RETURNS A NEW STATE

`(state, action) => newState`

ACTION

REDUCER

STORE

THE **STORE** HOLDS THE STATE OF THE APPLICATION, IT CALLS THE REDUCER WHEN ACTIONS ARE DISPATCHED

## CREATE A STORE WITH A REDUCER

```
store = Redux.createStore(reducer)
```

## GET STATE THROUGH STORE

```
store.getState()
```

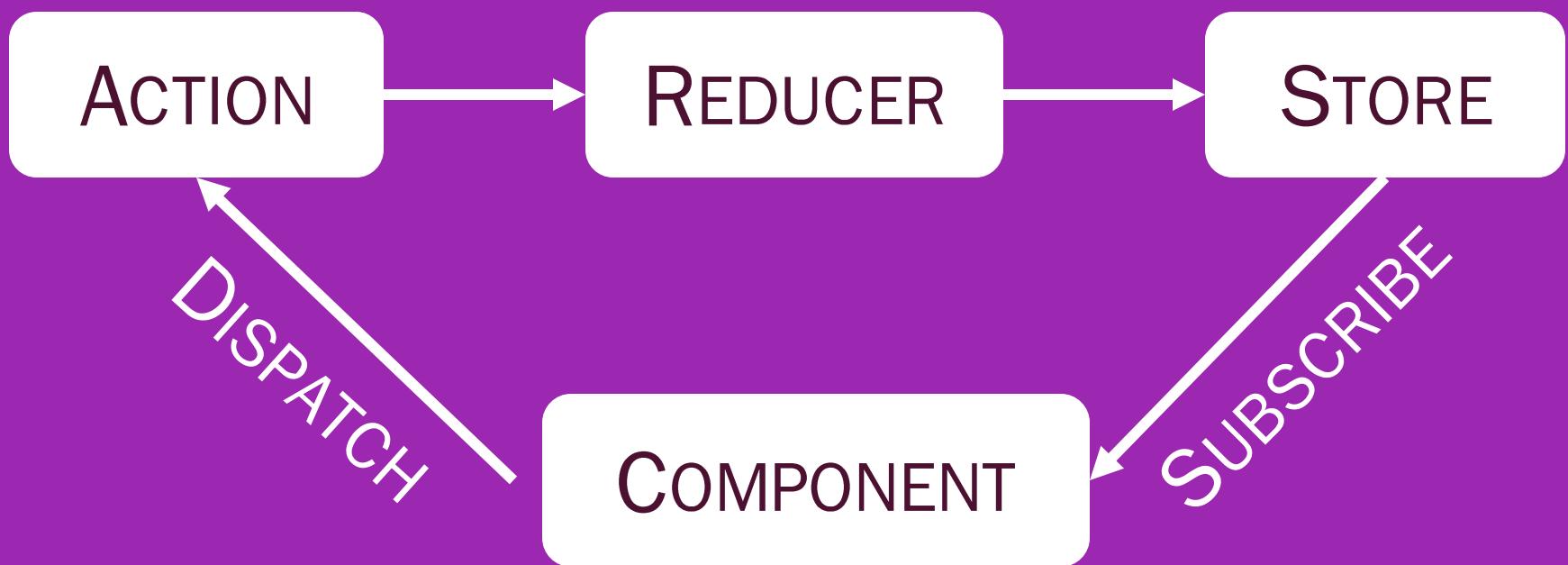
## DISPATCH ACTIONS THROUGH STORE

```
store.dispatch(addTodoAction())
```

## SUBSCRIBE TO CHANGES ON STORE

```
store.subscribe(() => {...})
```

$(\text{state}, \text{ action}) \Rightarrow \text{newState}$



# HELLO WORLD<sup>+</sup>

## LET'S GET STARTED!

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

## HELLO REAL WORLD

### WHAT'S NEXT?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

### SYNCING STATES

## HELLO REAL WORLD

### WHAT'S NEXT?

The screenshot shows a browser window with the title "Redux TodoMVC example" and the URL "zalmoxisus.github.io/redux-devtools-extension/examples/todomvc/". The page content is about "1. With Redux" and "1.1 Basic store". It includes a code snippet for creating a Redux store and a note about the optional `preloadedState` argument. Below the code, there's a section about ESLint configuration. On the left, the Redux DevTools extension is visible, showing a state tree with nodes like "state", "todos", and "ADD\_TODO". The "Todos" node has a value of 1, and the "ADD\_TODO" action has a "text" field of "Add extension". The bottom part of the screenshot shows the browser's address bar and some UI elements.

## 1. With Redux

### 1.1 Basic store

For a basic Redux store simply add:

```
const store = createStore(  
  reducer, /* preloadedState, */  
  + window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__()  
);
```

Note that `preloadedState` argument is optional in Redux' `createStore`.

For universal ("isomorphic") apps, prefix it with `typeof window !== 'undefined' &&`.

In case ESLint is configured to not allow using the underscore dangle, wrap it like so:

```
+ /* eslint-disable no-underscore-dangle */  
const store = createStore(  
  reducer, /* preloadedState, */  
  window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__()  
);
```

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

### SYNCING STATES

## HELLO REAL WORLD

### WHAT'S NEXT?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

#### SYNCING STATES

##### HELLO REAL WORLD

###### WHAT'S NEXT?

# UNDO/REDO REDUCER ENHancers!

A REDUCER ENHANCER IS  
SIMPLY A  
**HIGHER ORDER REDUCER**

# redux undo/redo

npm v0.6.1 build passing dependencies up to date code style standard tips \$0.00/week

*simple undo/redo functionality for redux state containers*

Clicked: 1 times



**Protip:** You can use the [redux-undo-boilerplate](#) to quickly get started with redux-undo.

**Note:** Make sure to update your programs to the [latest History API](#).

## Installation

```
npm install --save redux-undo
```

# UNDO/REDO REDUCER ENHancers!

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

#### SYNCING STATES

##### HELLO REAL WORLD

###### WHAT'S NEXT?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

# SYNCING STATE REDUX MIDDLEWARE!

MIDDLEWARE IS AN  
EXTENSION POINT  
BETWEEN THE  
ACTION DISPATCH AND  
THE REDUCER

# redux-localstorage-simple public

Save and load Redux state to and from LocalStorage. Supports Immutable.js data structures.

## Installation

```
npm install --save redux-localstorage-simple
```

## Usage Example (ES6 code)

```
import { applyMiddleware, createStore } from "redux"
import reducer from "./reducer"

// Import the necessary methods for saving and loading
import { save, load } from "redux-localstorage-simple"
```

# SYNCING STATE REDUX MIDDLEWARE!

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

HELLO REAL WORLD  
LESSONS LEARNED

# REDUX (WITH REACT) AT VISTAPRINT

Search Design

Enter a keyword

See your info  
all designs

Personalize it

Industry ▾

Art & Entertainment

Automotive & Tra  
(10)

Beauty & Spa (11)

Business Services

Construction, Rep  
Improvement (62)

Finance & Insuran

Health & Social Si

Law, Public Safet  
(18)

Retail & Sales (14)

Using Your Photo  
(20)

+ Show More

Personal & Family

Styles & Themes ▾

- + NO MORE COMPLEX PROPS TREE
- + SEPARATION OF CONCERNS
  - ADDING REDUX WAS A REWRITE
- + DEBUGGING AND ONBOARDING
  - DEBUGGING GENERATED STATE
  - LEARNING CURVE
- + UNIT TESTING/TDD

HELLO REAL WORLD  
LESSONS LEARNED

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

HELLO WORLD

REDUX DEV TOOLS

UNDO/REDO

SYNCING STATES

HELLO REAL WORLD

WHAT'S NEXT?



To use Redux, just remember the 3 parts:

- 1) Store:** the entire state of your app
- 2) Action:** description of what happened
- 3) Reducer:** a function that creates a new modified store based on the action



# WHAT is Angular Redux?



Angular Redux is an npm  
package that provides a  
series of Redux bindings  
for Angular



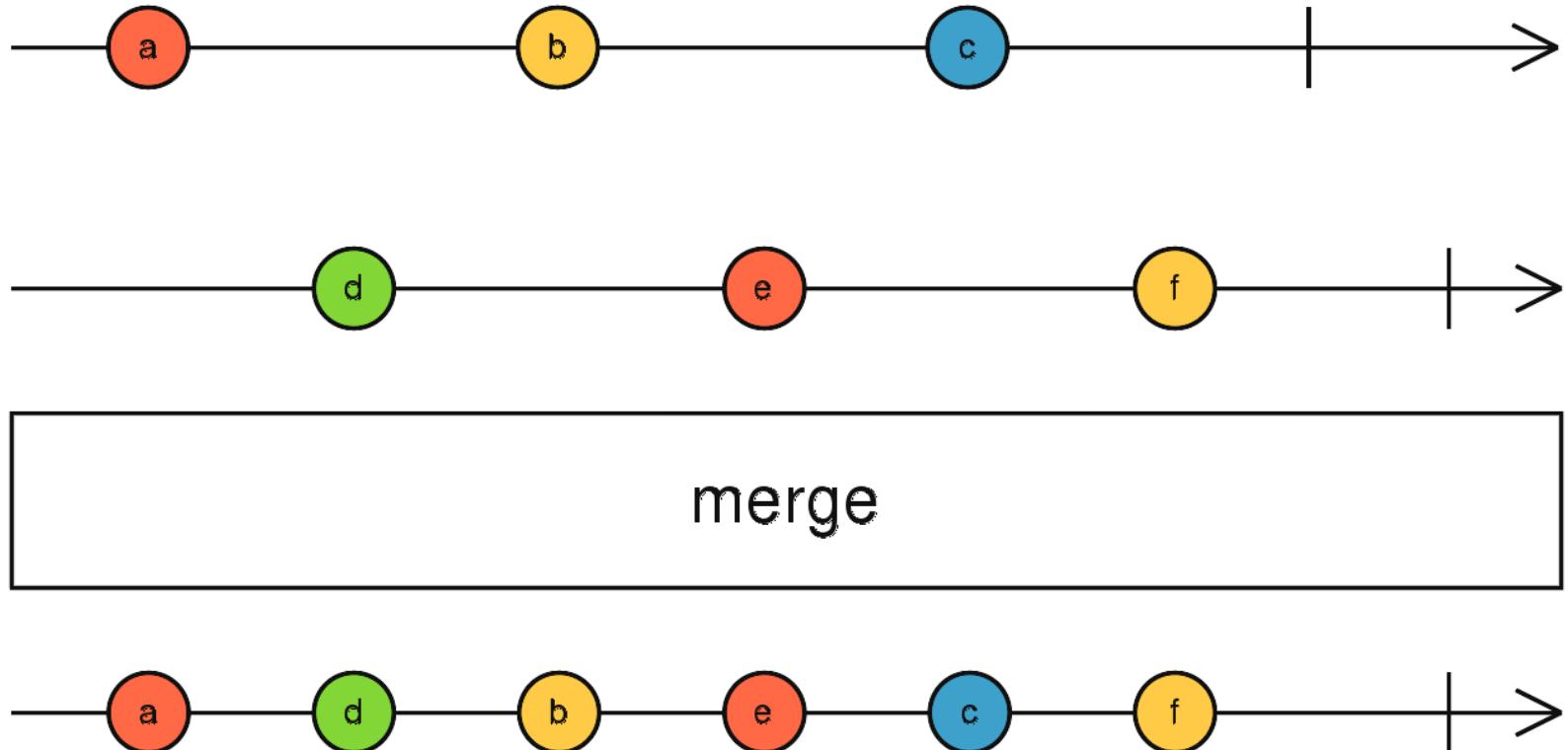
# HOW do I use Angular Redux?

# QUICK REVIEW

## Observables

# The Observable object represents a push based collection

The Observer and Observable interfaces provide a generalized mechanism for push-based notification, also known as the observer design pattern. The Observable object represents the object that sends notifications (the provider); the Observer object represents the class that receives them (the observer).



```
1 const sourceOne = Rx.Observable.create(observer => {
2   observer.onNext(1);
3   observer.onNext(2);
4   observer.onNext(3);
5 })
6 sourceOne.subscribe(val => console.log('SourceOne:', val));
7

8
9 const sourceTwo = Rx.Observable.interval(2000);
10 sourceTwo.subscribe(val => console.log('SourceTwo:', val));
```

```
1▼ const {Component} = ng.core;
2▼ const {bootstrap} = ng.platform.browser;
3
4▼ @Component({
5    selector: 'my-app',
6    template: `
7        <section>
8            <h1>{{number$ | async}}</h1>
9        </section>
10    `
11})
12▼ class AppComponent {
13▼   constructor(){
14      this.number$ = Rx.Observable.interval(1000);
15    }
16}
17
18bootstrap(AppComponent);
19
```



# HOW do I use Angular Redux?

HELLO WORLD

REDUX DEV TOOLS

UNDO/REDO

SYNCING STATES

HELLO REAL WORLD

WHAT'S NEXT?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

# HELLO WORLD<sup>+</sup>

## LET'S GET STARTED!

# @angular-redux/store

---

Angular bindings for [Redux](#).

For Angular 1 see [ng-redux](#)

[chat](#) [on gitter](#) [build](#) [passing](#) [npm](#) [v6.5.7](#) [downloads](#) [29k/month](#)

## What is Redux?

---

Redux is a popular approach to managing state in applications. It emphasises:

- A single, immutable data store.
- One-way data flow.
- An approach to change based on pure functions and a stream of actions.

You can find lots of excellent documentation here: [Redux](#).

## What is @angular-redux?

---

We provide a set of npm packages that help you integrate your redux store into your Angular 2+ application. This helps you by bridging the gap with some of Angular's advanced features, including:

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

## HELLO REAL WORLD

### WHAT'S NEXT?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

### SYNCING STATES

## HELLO REAL WORLD

### WHAT'S NEXT?

# REDUX DEVTOOLS

## YOUR BEST FRIEND!

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

### SYNCING STATES

## HELLO REAL WORLD

### WHAT'S NEXT?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

# SYNCING STATE REDUX MIDDLEWARE!

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

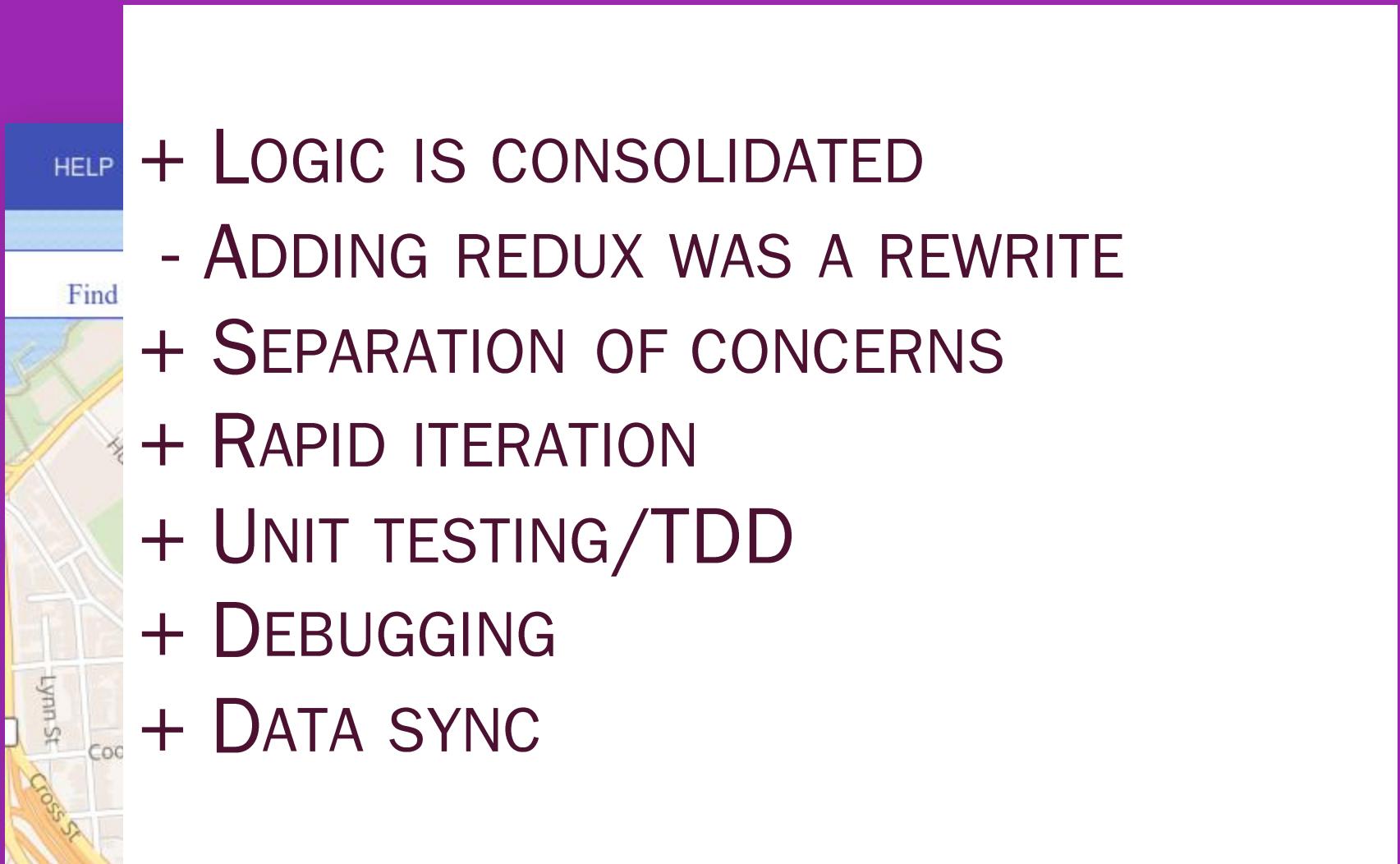
## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

HELLO REAL WORLD  
LESSONS LEARNED

# REDUX (WITH ANGULAR) AT PARKABLER



HELLO REAL WORLD  
LESSONS LEARNED

# HELLO WORLD

## REDUX DEV TOOLS

### UNDO/REDO

## SYNCING STATES

# HELLO REAL WORLD

## WHAT'S NEXT?

HELLO WORLD

REDUX DEV TOOLS

UNDO/REDO

SYNCING STATES

HELLO REAL WORLD

WHAT'S NEXT?



# To use Angular Redux:

- 1) npm install @angular-redux/store
- 2) Import/Setup your app
- 3) Store, Action, Reducer



# WHY should I use Angular Redux?



# Angular Redux is awesome because:

- Redux to manage your state
- Redux tools work out of the box
- Works with RxJS Observable
- ...



# Angular Redux

## WHAT is it ?

## HOW can I use it ?

## WHY should I care ?



# Is Angular Redux READY to use in Prod?

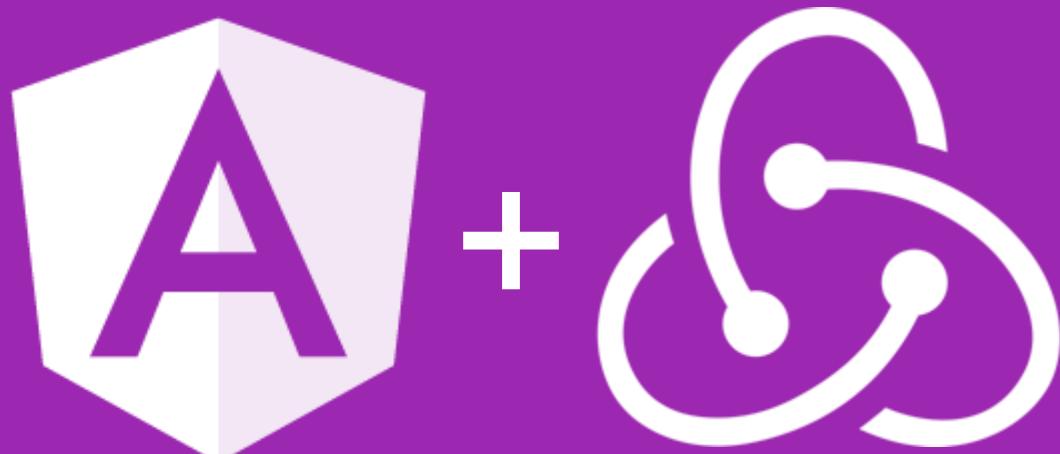


# Angular Redux is in 6.2.1

- Not an officially supported package by Google, open-source and maintained by Rangle.IO
- <https://github.com/angular-redux/store>

# THANK YOU!





MANAGE STATE

# Angular Redux

Christina Kayastha  
Senior Software Engineer  
Vistaprint, Cimpress  
@christikaes