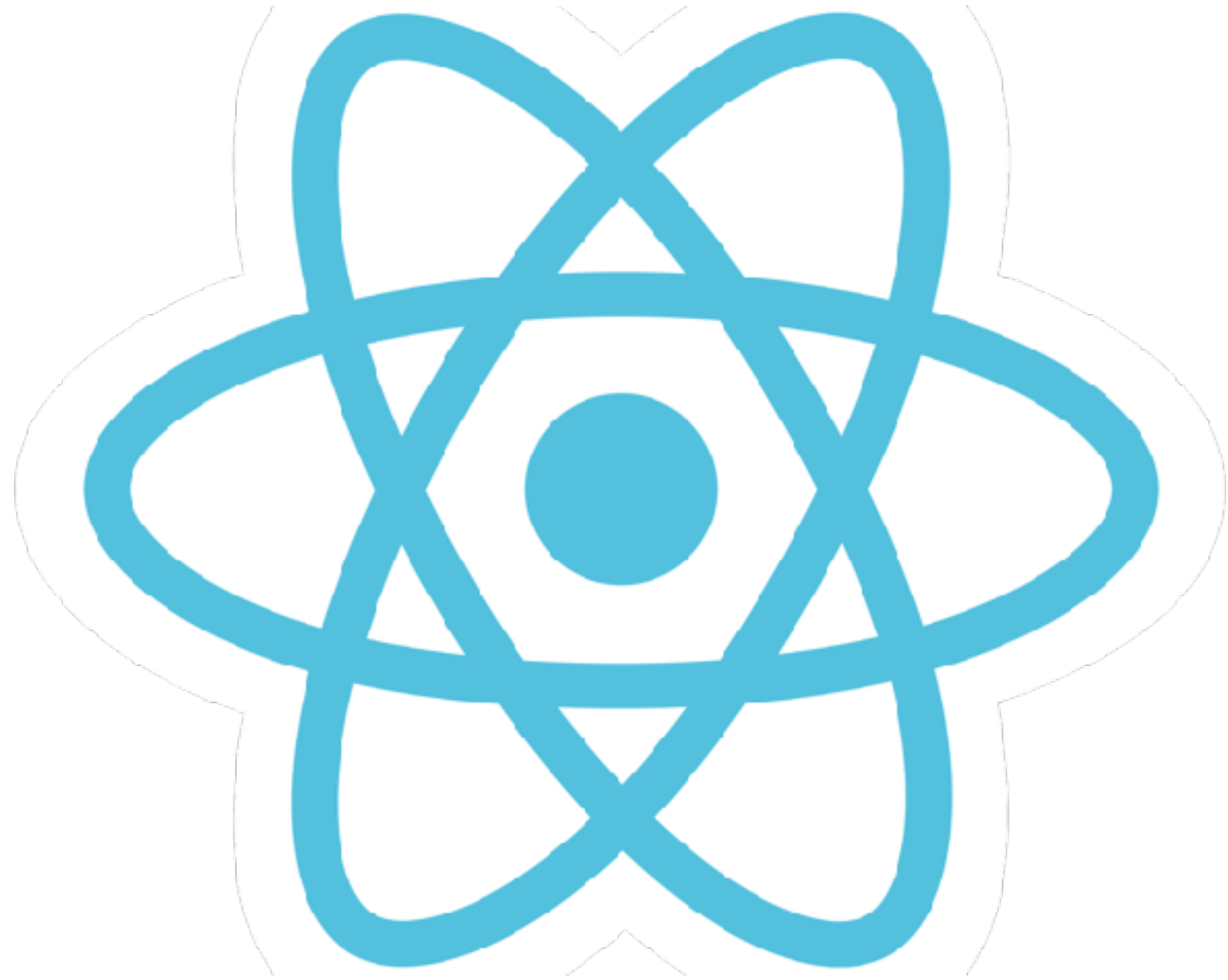# React Development

TekAdept, LLC   17, 18 & 24 Mar 2018

Introduction

Troy Miles

- Troy Miles

- Nearly 40 years of experience

- Programmer, speaker & author

- rockncoder@gmail.com

- @therockncoder

- lynda.com Author

Day One: React

- Introduction

- Setting Up

- Yarn

- Modern JavaScript

- Components & JSX

- React & the DOM

- Deeper JSX

- Jest

- Forms

- Debugging

Day Two:Routing & Redux

- Routing

- Parameters

- Programmatic Routes

- Animation

- Deploying to Firebase

- React Universal

- Redux

- ToDo with Redux

- Best Practices

Day Three: Redux

- Patterns

- Middleware

- DevTools

- Logger

- Sagas

- Generators

- Ajax & Sagas

- Helpers/Effects

- Testing Sagas

- Architecting Apps
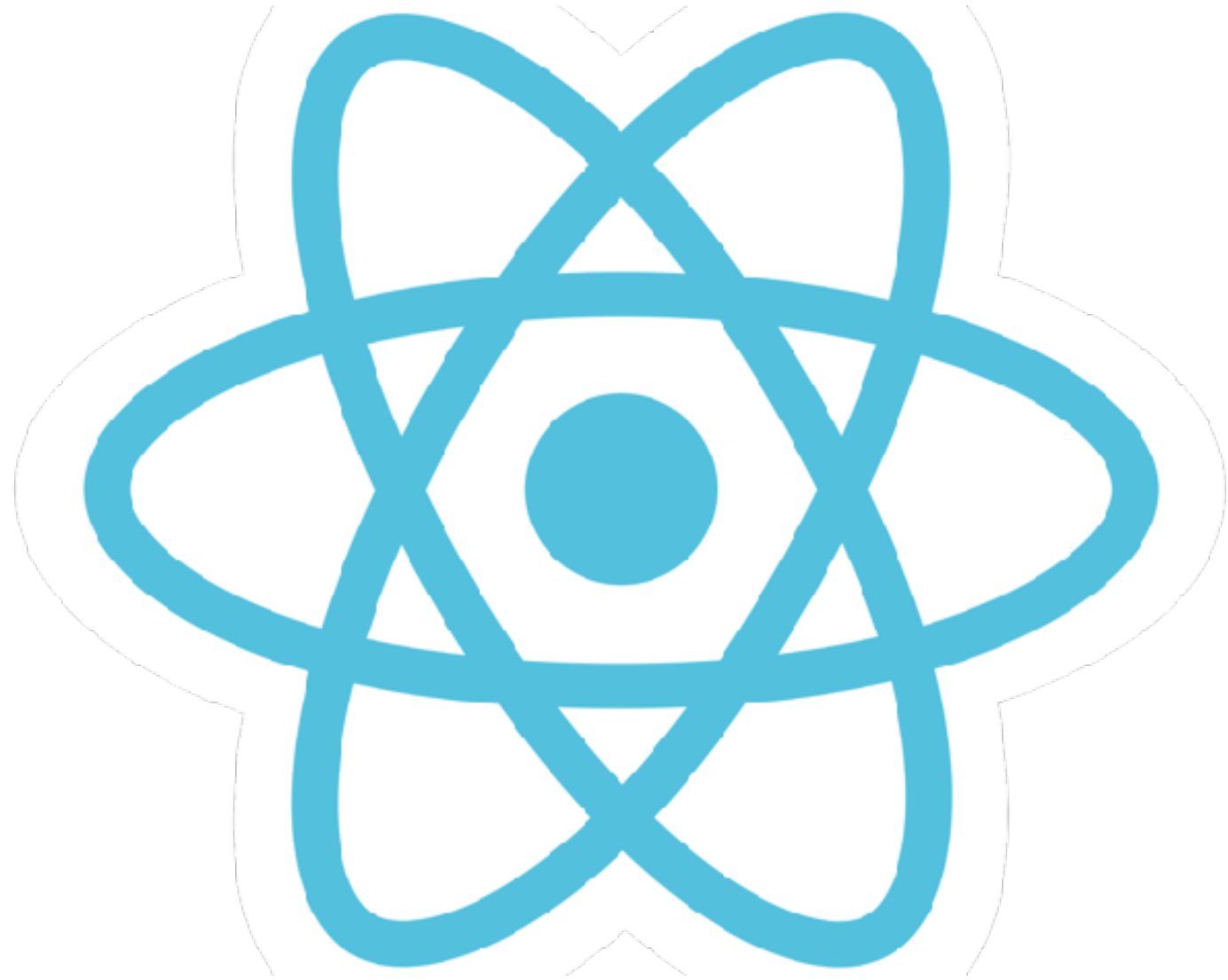
Setting Up

Prerequisites

- git

- Node.js/npm

- Yarn (npm install -g yarn)

installation

- npm install -g yarn

- yarn create react-app <app name>

- cd <app name>

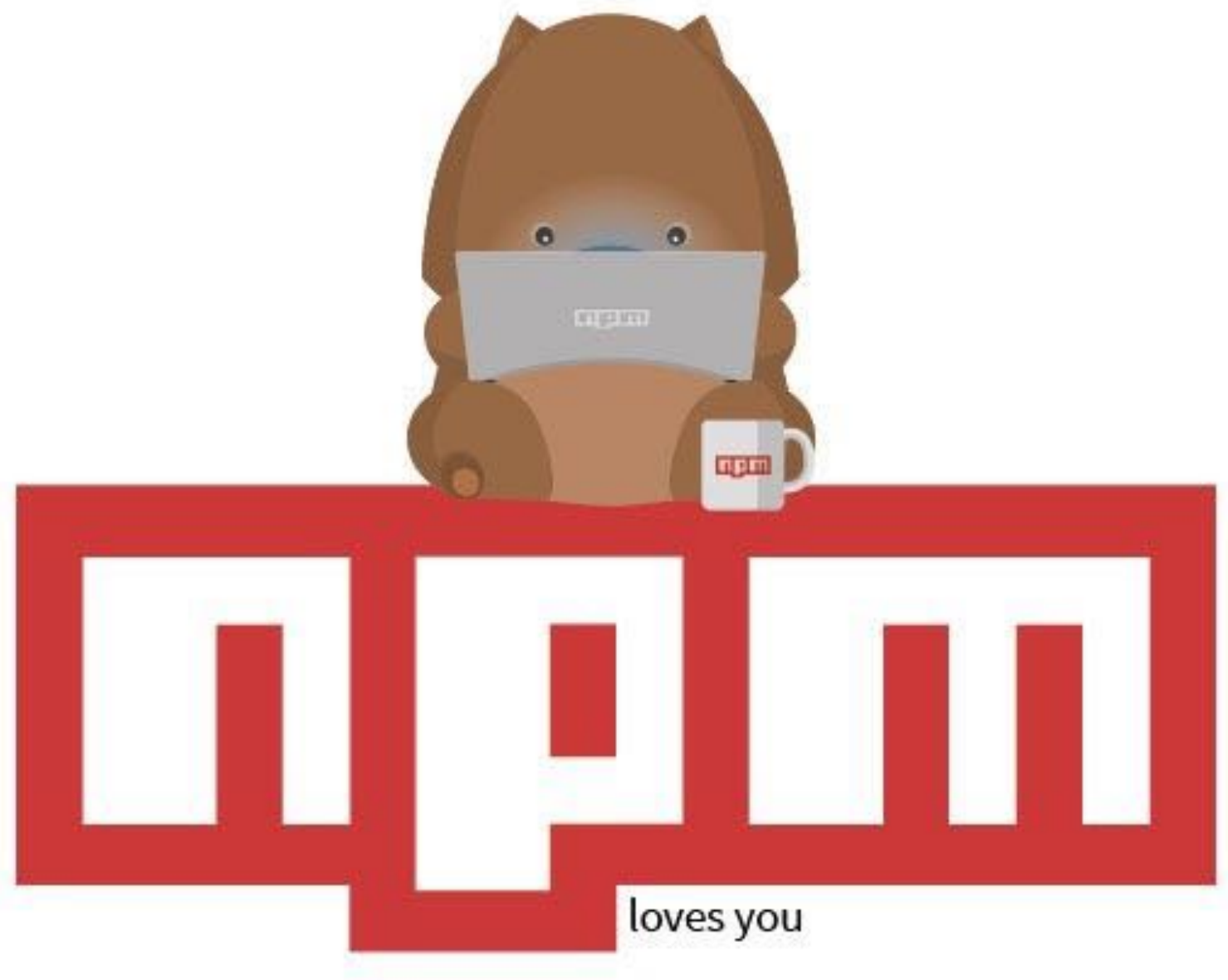- yarn start

Check Versions

app	command	my version
git	git —version	2.15.0
node.js	node -v	8.6.0
npm	npm —v	5.6.0
yarn	yarn -v	1.5.1
react	(React.version)	16.2.0

Node.js

Node.js

- Node.js is an open-source runtime for JavaScript

- Created by Ryan Dahl in 2009

- It allows JS code to run from the command line

- Can be used for web servers and CLI commands

npm

npm

- We need Node.js for its package manager, npm

- Created by Isaac Z. Schlueter in 2010

- NPM requires a file, package.json, in the app root

- We use npm for a lot of things

- https://www.npmjs.com/

www.npmjs.com

- This is the go to place to learn about packages

- Packages can be searched

- Instructions on how to install each package

- And interesting stats

yarn

yarn

- replaces npm and uses its infrastructure

- Create by Facebook, Exponent, Google, and Tilde

- Debut in October of 2016

- Concerns about npm's speed, security, and consistency

- Features: offline mode, deterministic, network performance

package.json

- name - what this app is called, for npm it must be

- version - the current version number

- description - what this package is all about

- author - who wrote it

- repository - where it lives

package.json

- license - type of license granted to users of the package

- scripts - commands

- dependencies - packages for production

- devDependencies - packages for development

version definitions

- "^1.8.9"

- 1 - is the major number

- 8 - is the minor number

- 9 - is the patch number

- patch number is not required

version symbols

- "1.8.9" - must exactly match version

- ">1.8.9" - must be greater than version

- ">=1.8.9", "<1.8.9", "<=1.8.9"

- "^1.8.9"- allows changes to the minor & patch

- "~1.8.9" - allows changes to the patch

- "1.8.*" - a wild card which stands for any value here

yarn commands

command	Purpose
yarn create react-app <>	Creates a react app
yarn add <>	Adds a package
yarn remove <>	Deletes a package
yarn start	Run app with hot-loading
yarn build	Creates a production build
yarn eject	Frees your app
yarn	Installs packages
yarn help	Lists all commands

What's in our package.json?

- react

- react-dom

- react-scripts

lock files

- package-lock.json / yarn.lock

- Specifies package install info :

  - version

  - URL of where retrieved
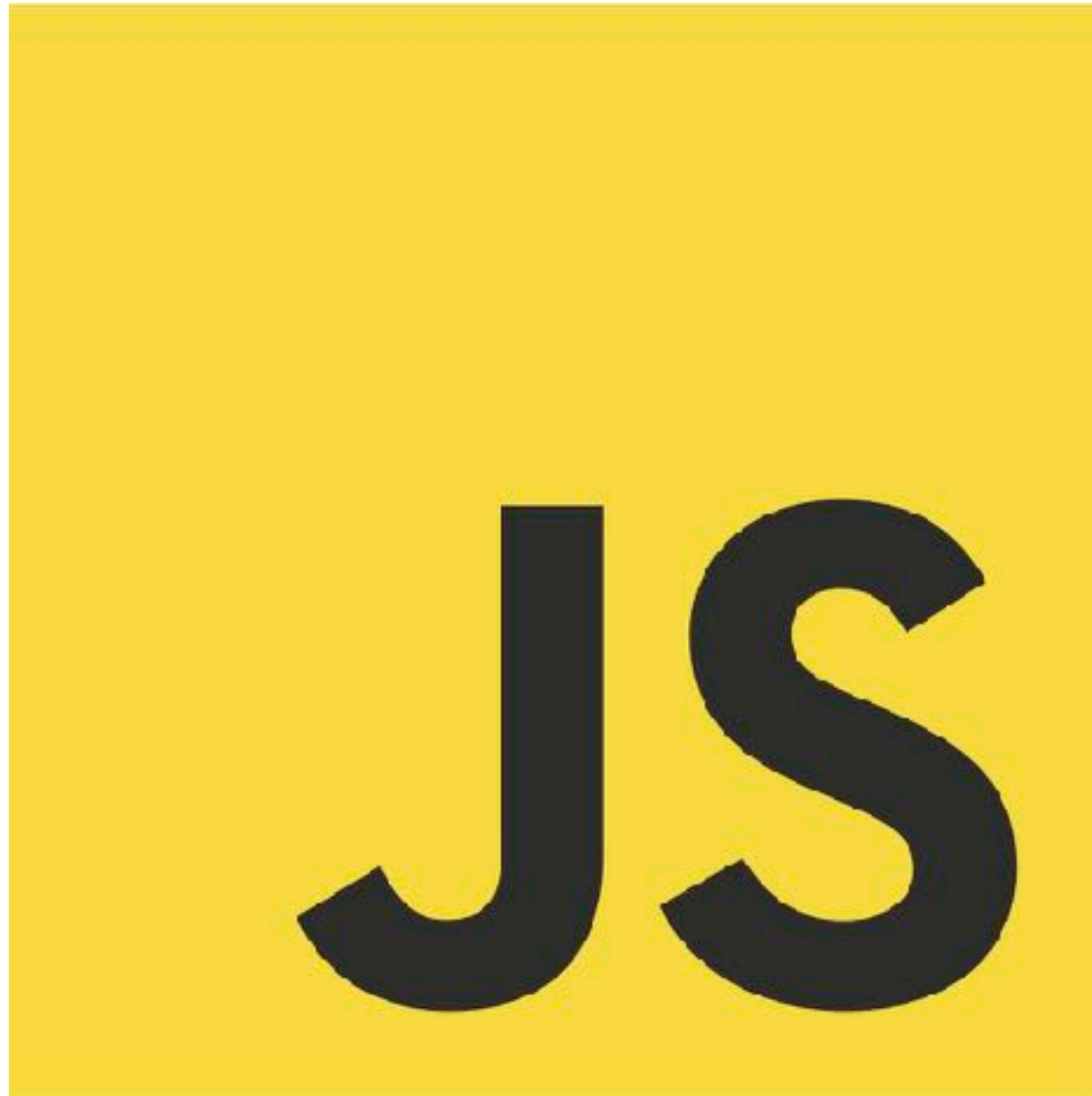
  - dependencies

- Don't edit it

Where can you look up npm info?

A. https://facebook.github.io/react/

B. https://www.npmjs.com/

C. http://redux.js.org/
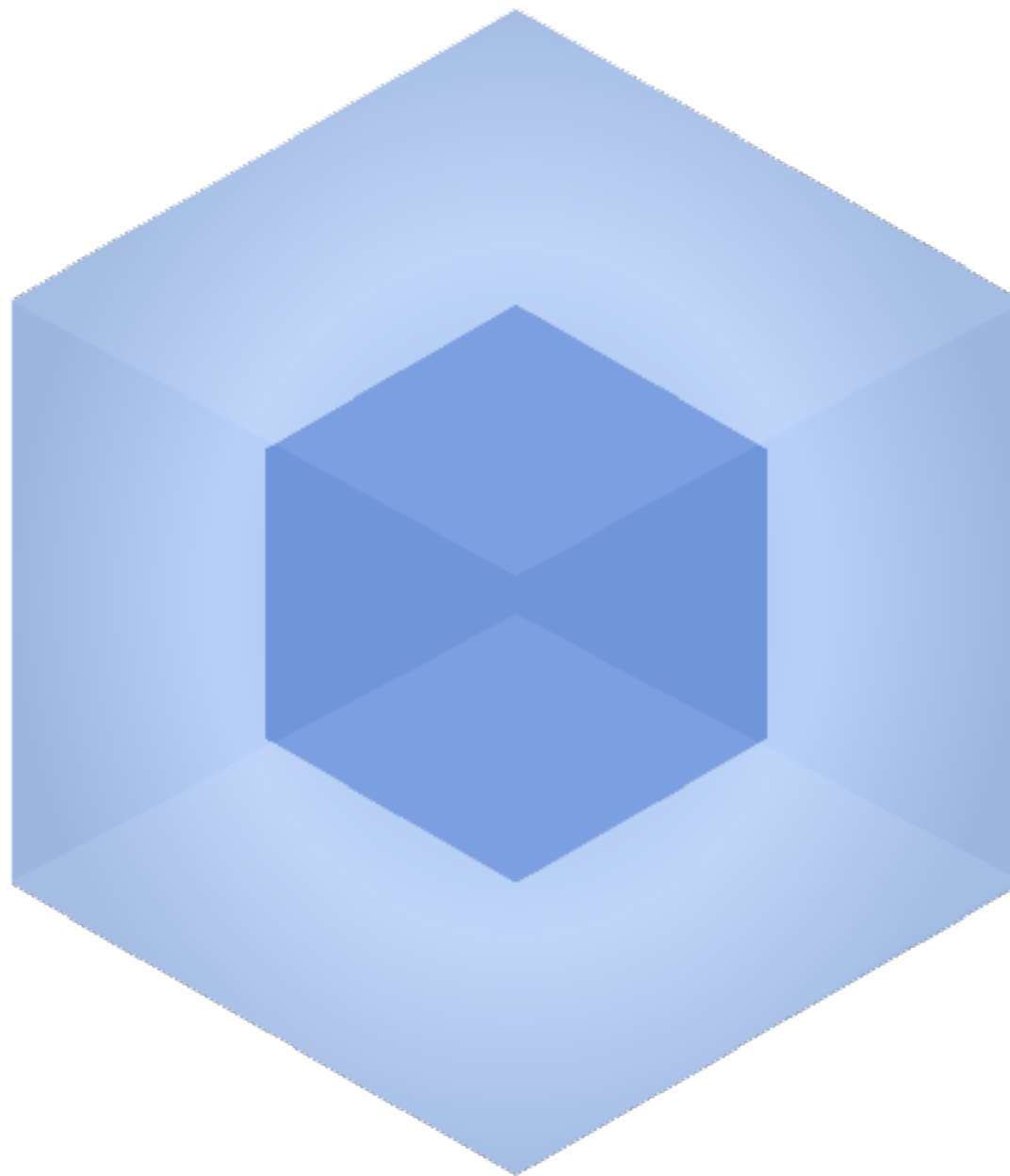
D. https://nodejs.org/

control-c

- To abort apps, press control-c

- Sometimes control-d or control-z will also work

Modern JavaScript

Webpack

Webpack

- Module bundler

- Created by Tobias Koppers & Sean Larkin in 2012

- Replaces System.JS, Browserify, and others

- Works with JS, CSS, and HTML

- Minifies, concatenates, and bundles

How?

- Webpack starts at your app's entry point

- It makes a graph of all of its dependencies

- It then bundles them together into an output file

- The graph is a DAG, directed acyclic graph

Loaders

- Goal: Webpack handler loading of all of your app's assets

- Every file is a module

- Webpack only understands only JavaScript

- Loaders transform files into modules

Rules

- test: Identifies the file's type

- use: Transform the file with a plugin loader

Example #1

```
module: {
  rules: [
    {
      test: /\.jsx*$/,
      exclude: [/node_modules/, /.+\.config.js/],
      loader: 'babel-loader',
    },
```

Example #2

```
{
  test: /\.css$/,
  use: [
    {
      loader: 'style-loader',
      options: {
        sourceMap: true
      }
    },
    {
      loader: 'css-loader',
      options: {
        modules: true,
        importLoaders: 1,
        localIdentName: '[path]___[name]__[local]___[hash:base64:5]'
      }
    },
    {
      loader: 'postcss-loader'
    }
  ],
},
```

Before Bundling

```
14    componentWillMount() {
15      const app = this.props.db.database().ref(FirebaseTable);
16      app.on('value', snapshot ⇒ {
17        this.getData(snapshot.val());
18      });
19    }
20
21    getData(values) {
22      if(values) {
23        const chats = Object.keys(values)
24          .map(key ⇒ Object.assign({}, values[key], {key}));
25        this.setState({chats});
26      }
27    }
```

After Bundling

```
.create(t&&t.prototype,{constructor:{value:e,enumerable:!1,writable:!0,configurable:!0}}),t&&(Object
.setPrototypeOf?Object.setPrototypeOf(e,t):e.__proto__=t)}var a=n(9),s=n.n(a),u=n(84),c=n(189),l=function()
{function e(e,t){for(var n=0;n<t.length;n++){var r=t[n];r.enumerable=r.enumerable||!1,r.configurable=!0,
"value"in r&&(r.writable=!0),Object.defineProperty(e,r.key,r)}}return function(t,n,r){return n&&e(t
.prototype,n),r&&e(t,r),t}}(),h=function(e){function t(e){r(this,t);var n=i(this,(t.__proto__||Object
.getPrototypeOf(t)).call(this,e));return n.state={chats:[]},n}return o(t,e),l(t,[{key:"componentWillMount",
value:function(){var e=this;this.props.db.database().ref(u.a).on("value",function(t){e.getData(t.val())})}},
{key:"getData",value:function(e){if(e){var t=Object.keys(e).map(function(t){return Object.assign({},e[t],
{key:t})});this.setState({chats:t})}}},{key:"render",value:function(){var e=this,t=this.state.chats.map
(function(t){return s.a.createElement("div",{className:"card",key:t.key},s.a.createElement("div",
{className:"card-content"},s.a.createElement(c.a,{chat:t.text,mine:e.props.guid===t.guid})))});return
s.a.createElement("div",null,t)}}]),t}(a.Component);t.a=h},function(e,t,n){"use strict";function r(e){return
o.a.createElement("div",{className:""},o.a.createElement("div",{className:e.mine?"has-text-right":""},e
.chat))}t.a=r;var i=n(9),o=n.n(i)},function(e,t,n){"use strict";function r(e,t){if(!(e instanceof t))throw
new TypeError("Cannot call a class as a function")}function i(e,t){if(!e)throw new ReferenceError("this
```

Babel

Babel

- The compiler for writing the next generation JavaScript

- current version 6.26.0

- works better with Webpack

Babel

- It is modular, with a small lightweight core

- Functionality comes from plugins

- All plugin are opt-in

Presets

- You might need a dozen plugins

- Keep track of them would be burdensome

- Presets are groups of related plugins

- Two popular ones are babel-preset-es2015 and babel-preset-react

yarn create react-app

- yarn create react-app <app name>

- Webpack is configured by react-scripts

- Look in node_modules/react-scripts/config

- Configurations for dev and prod builds are there
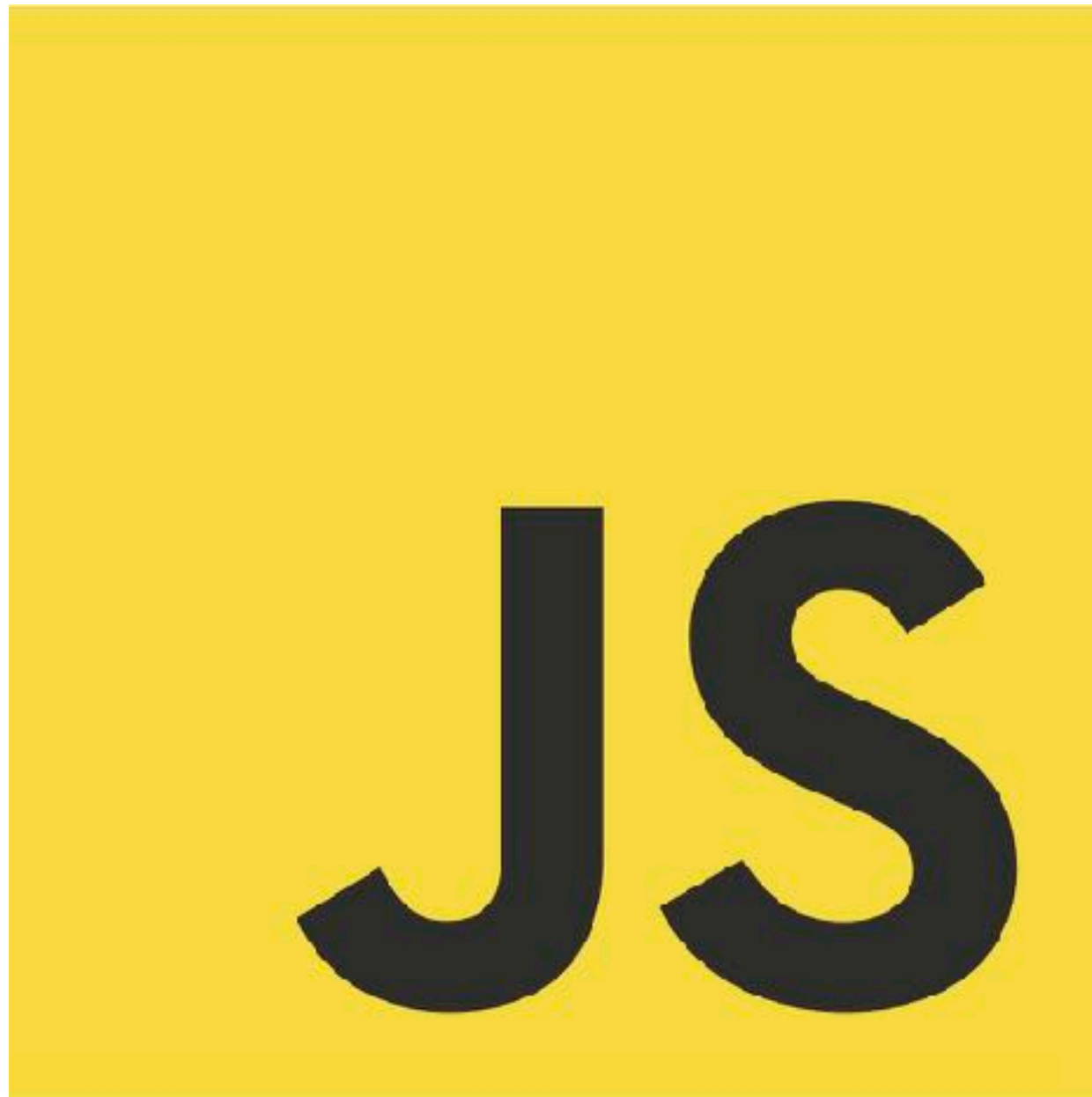
JavaScript

JavaScript

- A high-level interpreted programming language

- Created by Brendan Each in 1995

- Dynamic, weakly typed, prototype-based, & multi-paradigm

- Current version ES2017

ECMAScript Versions

Version	Date
ES1	June 1997
ES2	June 1998
ES3	December 1999
ES4	DOA 2006
ES5	December 2009
ES2015	June 2015 (ES6)
ES2016	June 2016
ES2017	June 2017

Collection Operators

- .isArray()

- .every()

- .forEach()

- .indexOf()

- .lastIndexOf()

- .some()

- .map()

- .reduce()

- .filter()

map

```
let junk = [1, 2, 3, 4, 'Alpha', 5, {name: 'Jason'}];
let letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K'];
let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];
console.log(nums);

// map iterates over all of the elements and returns a new array with the same
// number of elements
let nums2 = nums.map((elem) => elem * 2);
console.log(nums2);
/// [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40]
```

filter

```
let junk = [1, 2, 3, 4, 'Alpha', 5, {name: 'Jason'}];
let letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K'];
let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];
console.log(nums);

// filter iterates over the array and returns a new array with only the elements
// that pass the test
let nums3 = nums.filter((elem) => !!(elem % 2));
console.log(nums3);
/// [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

reduce

```
let junk = [1, 2, 3, 4, 'Alpha', 5, {name: 'Jason'}];
let letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K'];
let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];
console.log(nums);

// reduce iterates over the array passing the previous value and the current
// element it is up to you what the reduction does, let's concatenate the strings
let letters2 = letters.reduce((previous, current) => previous + current);
console.log(letters2);
/// ABCDEFGHIJK

// reduceRight does the same but goes from right to left
let letters3 = letters.reduceRight((previous, current) => previous + current);
console.log(letters3);
/// KJIHGFEDCBA
```

let

- let allows us to create a block scoped variables

- they live and die within their curly braces

- var is considered deprecated

- best practice is to use let instead of var

let

```javascript
// let allows us to create block scoped variables
// they live and die within the curly braces
let val = 2;
console.info(`val = ${val}`);
{
  let val = 59;
  console.info(`val = ${val}`);
}
console.info(`val = ${val}`);
```

const

- const creates a variable that can't be changed

- best practice is to make any variable that should not change a constant

- does not apply to object properties or array elements

const

```
const name = 'Troy';
console.info(`My name is ${name}`);
// the line below triggers a type error
name = 'Miles';
```

Template strings

- Defined by using opening & closing back ticks

- Templates defined by ${JavaScript value}

- The value can be any simple JavaScript expression

- Allows multi-line strings (return is pass thru)

Template strings

```
let state = 'California';
let city = 'Long Beach';
console.info(`This weekend's workshop is in ${city}, ${state}.`);

// template strings can run simple expressions like addition
let cup_coffee = 4.5;
let cup_tea = 2.5;
console.info(`coffee: $${cup_coffee} + tea: $${cup_tea} = $$
{cup_coffee + cup_tea}.`);

// they can allow us to create multi-line strings
console.info(`This is line #1.
this is line #2.`);
```

Arrow functions

- Succinct syntax

- Doesn't bind its own *this*, *arguments*, or *super*

- Facilitate a more functional style of coding

- Can't be used as constructors

Arrow functions

- When only one parameter, parenthesis optional

- When zero or more than one parameter, parenthesis required

Arrow function

```
let anon_func = function (num1, num2) {
  return num1 + num2;
};
console.info(`Anonymous func: ${anon_func(1, 2)}`);

let arrow_func = (num1, num2) => num1 + num2;
console.info(`Arrow func: ${arrow_func(3, 4)}`);
```

this

- this is handled different in arrow functions

- In anonymous function this is bound to the global object

- In arrow function this is what it was in the outer scope

Arrow Best Practices

- Use function in the global scope and for Object.prototype properties

- Use arrow functions everywhere else

Destructuring

- Maps the data on the right side of the equals sign to the variables on the left

- The data type decides the way values are mapped

- It is either object or array destructuring

Object Destructuring

```
16 // this is a demo of the power of destructuring
17 // we have two objects with the same 3 properties
18 const binary = {kb: 1024, mb: 1048576, gb: 1073741824};
19 const digital = {kb: 1000, mb: 1000000, gb: 1000000000};
20 // We use a ternary statement to choose which object
21 // assign properties based on their property names
22 const {kb, mb, gb} = (useBinary) ? binary : digital;
```

Array Destructuring

```
5
6    let [param1, bob, key] = ['first', 'second', '3rd'];
7    console.info(`param1 = ${param1}, bob = ${bob}, key = ${key}`);
8    // param1 = first, bob = second, key = 3rd
```

Spread syntax

- Expands an expression in places where multiple arguments, elements, or variables are expected

The spread operator

```javascript
11
12   // the spread operator
13   const myArray = ['Bob', 'Sue', 'Fido'];
14   function printFamily(person1, person2, pet) {
15     console.info(`Person 1: ${person1}, Person 2: ${person2}, and their pet: $
{pet}`);
16   }
17   printFamily(...myArray);
18   // Person 1: Bob, Person 2: Sue, and their pet: Fido
19
```

Generator Function*

- function* is a called a generator

- A generator is a function which can exited and re-entered

- statements are executed until a yield is encounter

- the generator then pauses execution until the yield returns

- if a return statement is encounter, it finishes the generator

Yield

- yield is a keyword which pauses the execution of a generator function

- yield can return a value to the generator

Class

- Syntactic sugar over JavaScript use of function constructors

- JavaScript uses proto-typical inheritance

- If a class **extends** another, it must include **super()** as first instruction in its **constructor**

- Only create a constructor if it does something

Class

- Syntactic sugar over JavaScript use of function constructors

- JavaScript uses proto-typical inheritance

- If a class **extends** another, it must include **super()** as first instruction in its **constructor**

- Only create a constructor if it does something

Object

- By itself creates an object wrapper

- Has several useful methods

- Object.assign copies an object

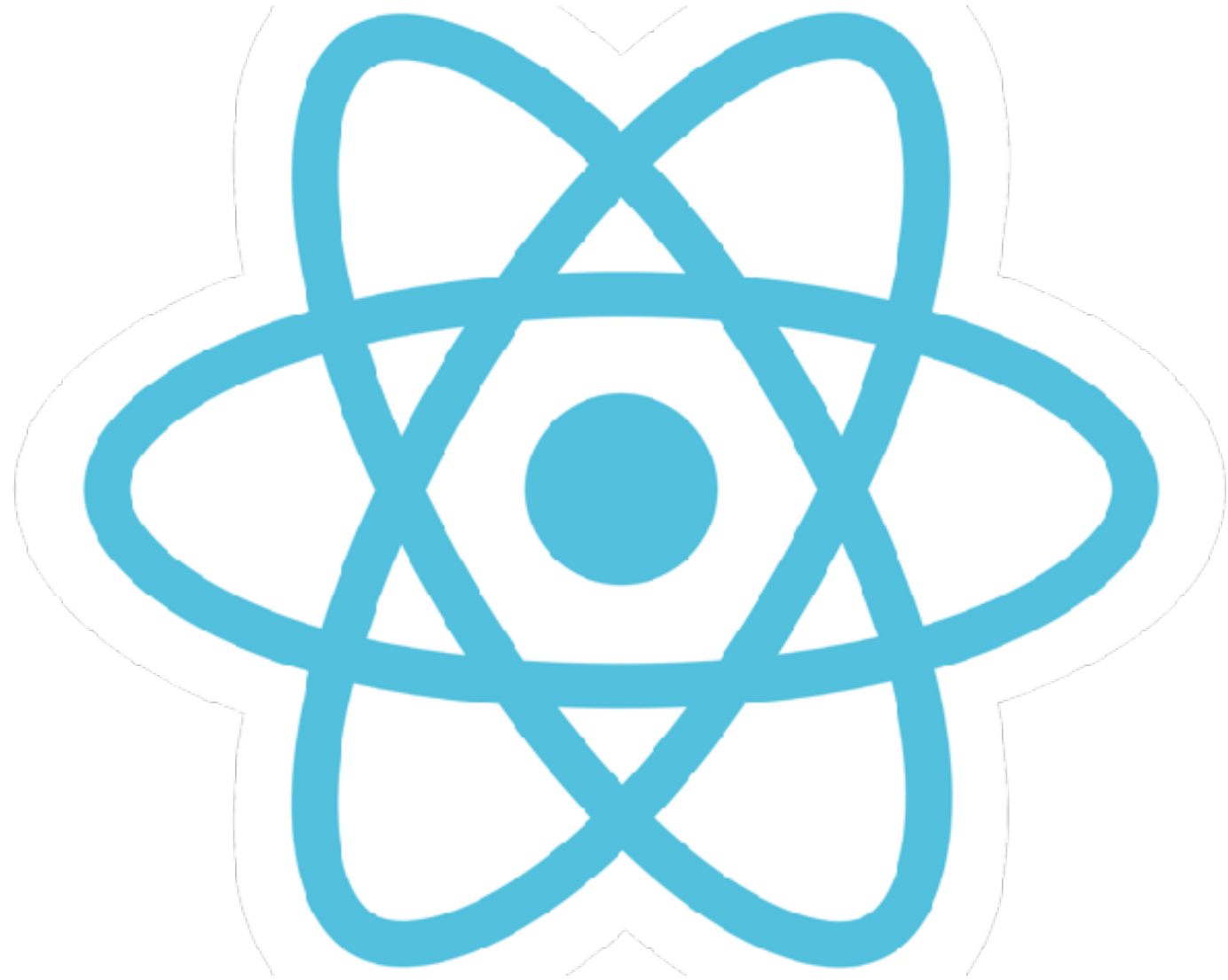- Object.keys creates an array of the object's keys

import

- Imports functions, objects, or primitives from other files

- import <name> from "<module name>";

- import {name } from "<module name>";

- import * as Greetings from "<module name>";

- relative path indicates not an npm package

export

- export <var a>

- export {a, b};

export default

- only one per file

- common pattern for libraries

- const Greetings = {sayHi, sayBye};

- export default Greetings;

- export default {sayHi, sayBye};

Components & JSX

React

- A JavaScript library for building user interfaces

- Created by Facebook & Instagram

- Initial release March 2013

- Current version 16.2.0, November 28, 2017

React

- Virtual DOM

- One-way data flow

- JSX - JavaScript eXtension allows in HTML generation

- Component-based

React API

- The use of JSX is optional in React

- You can use the React "API" instead

- *The createClass method has been removed from React 16**

Component

- Fundamental building block of React

- Can be created with a JS Class or Function

- The render method is mandatory

Class Component

```
3
4    class Square extends React.Component {
5      render() {
6        return (
7          <button
8            className="square"
9            onClick={() ⟹ this.props.onClick()}>
10           {this.props.value}
11         </button>
12       );
13     }
14   }
15
```

Functional Component

```
16    function Square(props) {
17      return (
18        <button
19          className="square"
20          onClick={() ⇒ props.onClick()}>
21          {props.value}
22        </button>
23      );
24    }
```

Props

- Props are read-only objects

- Passed to component via attributes from parent

- Access via *this.props*

State

- State is internal to the component

- It is mutable

- Access via *this.state*

- Initialize in the constructor

- Should only modified via setState

PropTypes

- PropTypes insure that props are passed correctly

- They are in the "prop-types" npm package

- `import PropTypes from 'prop-types';`

- *React.PropTypes is deprecated*

PropTypes

- PropTypes allow you to declare what properties your component expects

- React validates property at runtime

- Using propTypes is optional

Some PropTypes

Component	Command
PropTypes.array	an optional array
PropTypes.bool	an optional bool
PropTypes.element	An optional React element
PropTypes.func	An optional function
PropTypes.node	Anything that can be rendered
PropTypes.number	An optional number
PropTypes.object	An optional object
PropTypes.string	An optional string
PropTypes.symbol	An optional Symbol

PropType in code

```
 1
 2   import React, { Component } from 'react';
 3   import ReactDOM from 'react-dom';
 4   import PropTypes from 'prop-types';
 5
 6   class Name extends Component {
 7     render () {
 8       return React.DOM.span(null, `My name is ${this.props.name}`);
 9     }
10   }
11
12   Name.propTypes = {
13     name: PropTypes.string.isRequired
14   };
15
16   ReactDOM.render(
17     React.createElement(Name, {}),
18     document.getElementById('root')
19   );
20
21
```
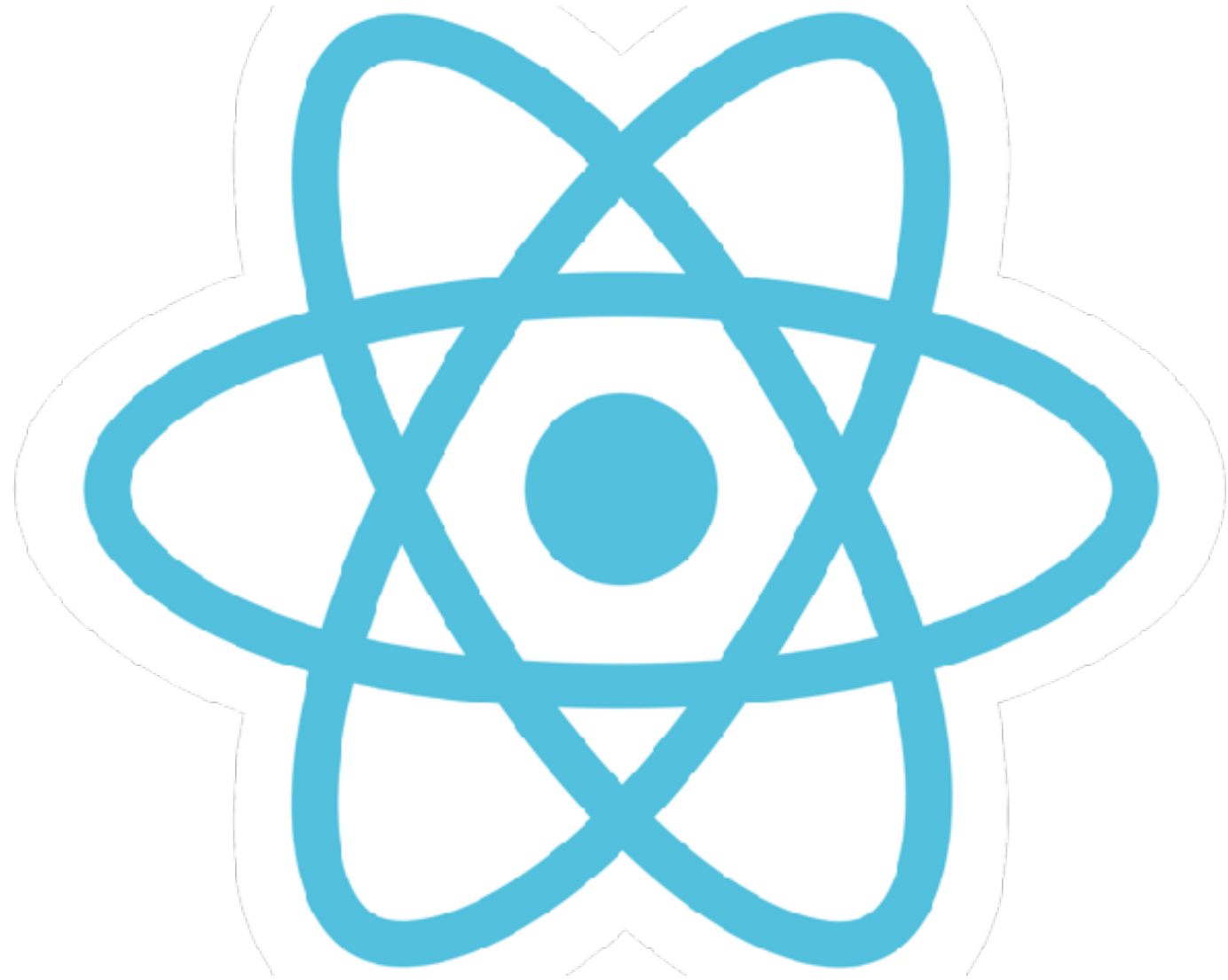
When property is missing

React Components

- Can be created two ways:

  - Using JavaScript

  - Using JSX

Components via JS

- React's API contains method createElement()

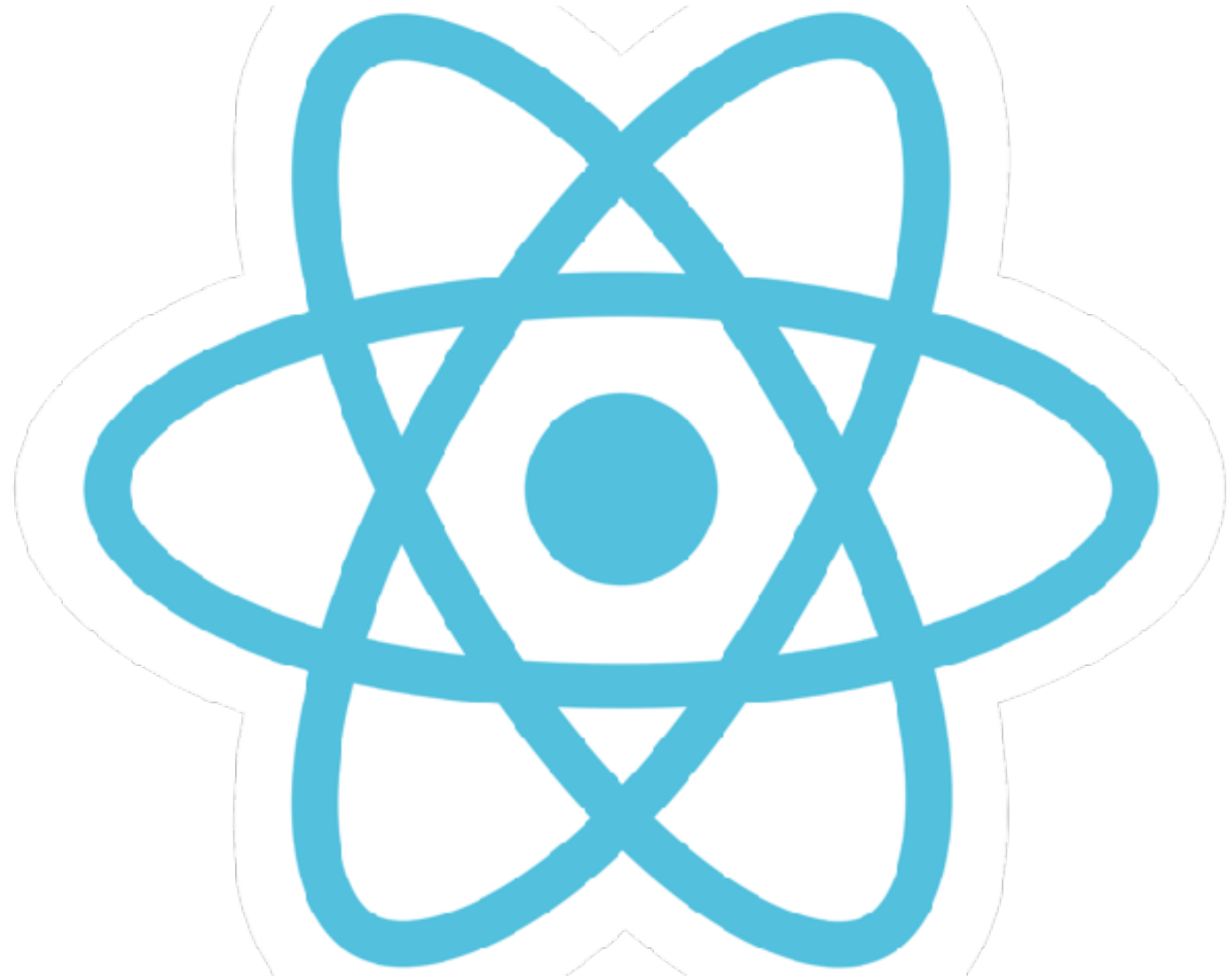- Takes 3 parameters: type, props, children

```
render() {

  return React.createElement('h1', null, "Hello there.");
```

React & the DOM

React & the DOM

- Events

- Hands-on: Event Handling

- Attributes

- Hands-on: Attribute passing

Deeper JSX

Deeper JSX

- Conditionals

- Comments

- Hands-on: Conditionals

- Element Factories

- Styling

- Hands-on: JSX & Styling

JSX

- **J**avaScript **S**yntax E**X**tension

- A mixture of JavaScript and HTML syntax

- Compiles to JavaScript

- Is optional but preferred over using JavaScript

- Is easier to read

JSX Attributes

- Can assign either a string or an expression

- Strings can be either single or double quotes

- Expressions must be enclosed with curly braces

Boolean Attributes

- HTML has a few boolean attributes, if present they are true

- Some of these include: checked, selected, disabled

- In JSX, <Component disabled={true / false} />

Forbidden Attributes

- Two attributes are JavaScript keywords

- JavaScript keywords can't be used in JSX

  - class -> className

  - for -> htmlFor

JSX Spread Syntax
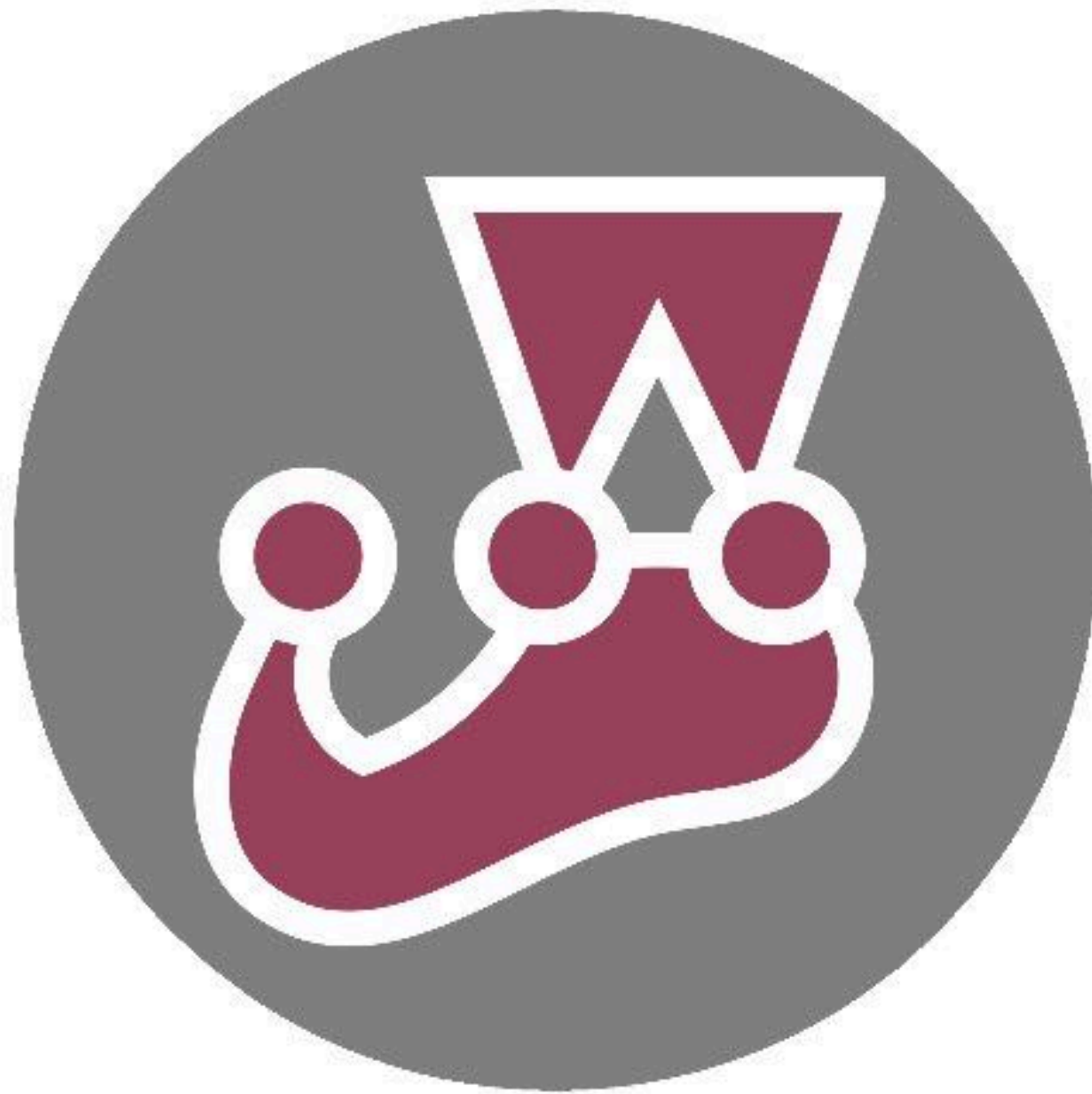
- a shortcut to passing props to a component

- uses ES2015 spread operator

- <Component {…object} />

JSX Spread Syntax

```
return (
    <Timer
            id={this.props.id}
            amount={this.props.amount}
            elapsed={this.props.elapsed}
            runningSince={this.props.runningSince}
            onStartClick={this.props.onStartClick}
            onStopClick={this.props.onStopClick}
            onResetClick={this.props.onResetClick}
            onSetClick={this.handleSetClick}
    />
);
```

JSX Spread Syntax

```jsx
return (
  <Timer
        { ... this.props}
        onSetClick={this.handleSetClick}
  />
);
```

Jest

Jest

- Created by Facebook in 2014

- Current version 22.4

- The default unit test for React

- Can test client and server code

describe() - Test Suite

- describe() is a global Jest function

- Takes two parameters

  - name of the test suite (string)

  - implementation of the test suite (function)

- Can be nested

it() - Test Spec

- it() is a global Jest function

- Takes two parameters

  - name of the spec (string)

  - implementation of the spec (function)

test() - Test Spec

- test() is a global Jest function

- Takes three parameters

  - name of the spec (string)

  - implementation of the spec (function)

  - *optional* timeout in milliseconds

expect()

```
expect(app).toBeTruthy();
expect(app).not.toBeUndefined();
expect(app.title).toEqual('Quizzer');
```

Matcher

- Takes the output of the expect() function

- Takes one parameter

  - The expected value

- Compares the expect and actual value using the logic of the matcher

Matchers (part 1)

Matcher	Comparison
toBe()	check exact equality
toBeCloseTo()	comparing floating point numbers
toBeDefined()	compares against 'undefined'
toBeFalsy()	falsy boolean casting
toBeGreaterThan()	mathematical comparison
toBeGreaterThanOrEqual()	mathematical comparison

Matchers (part 2)

Matcher	Comparison
toBeInstanceOf()	check if instance of a class
toBeLessThan()	mathematical comparison
toBeLessThanOrEqual()	mathematical comparison
toBeNull()	same as toBe(null)
toBeTruthy()	truthy boolean casting

Matchers (part 3)

Matcher	Comparison
toBeUndefined()	compares against undefined
toContain()	finds an item in array
toContainEqual()	finds an item in array exactly
toEqual()	works for literal variables and objects
toHaveBeenCalled()	ensure a mock function got called

Matchers (part 4)

Matcher	Comparison
toHaveBeenCalledTimes()	called exact number of times
toHaveBeenCalledWith()	called with specific args
toHaveBeenLastCalledWith()	called the last time with args
toHaveLength()	object length matches
toHaveProperty()	object property matches

Matchers (part 5)

Matcher	Comparison
toMatch()	for regular expressions
toMatchObject()	
toMatchSnapshot()	for snapshots
toThrow()	should throw an exception
toThrowErrorMatchingSnapshot()	

.not.

- Reverses the logic of the matcher

beforeEach()

- Setup function

- Called before each spec is executed

afterEach()

- Teardown function

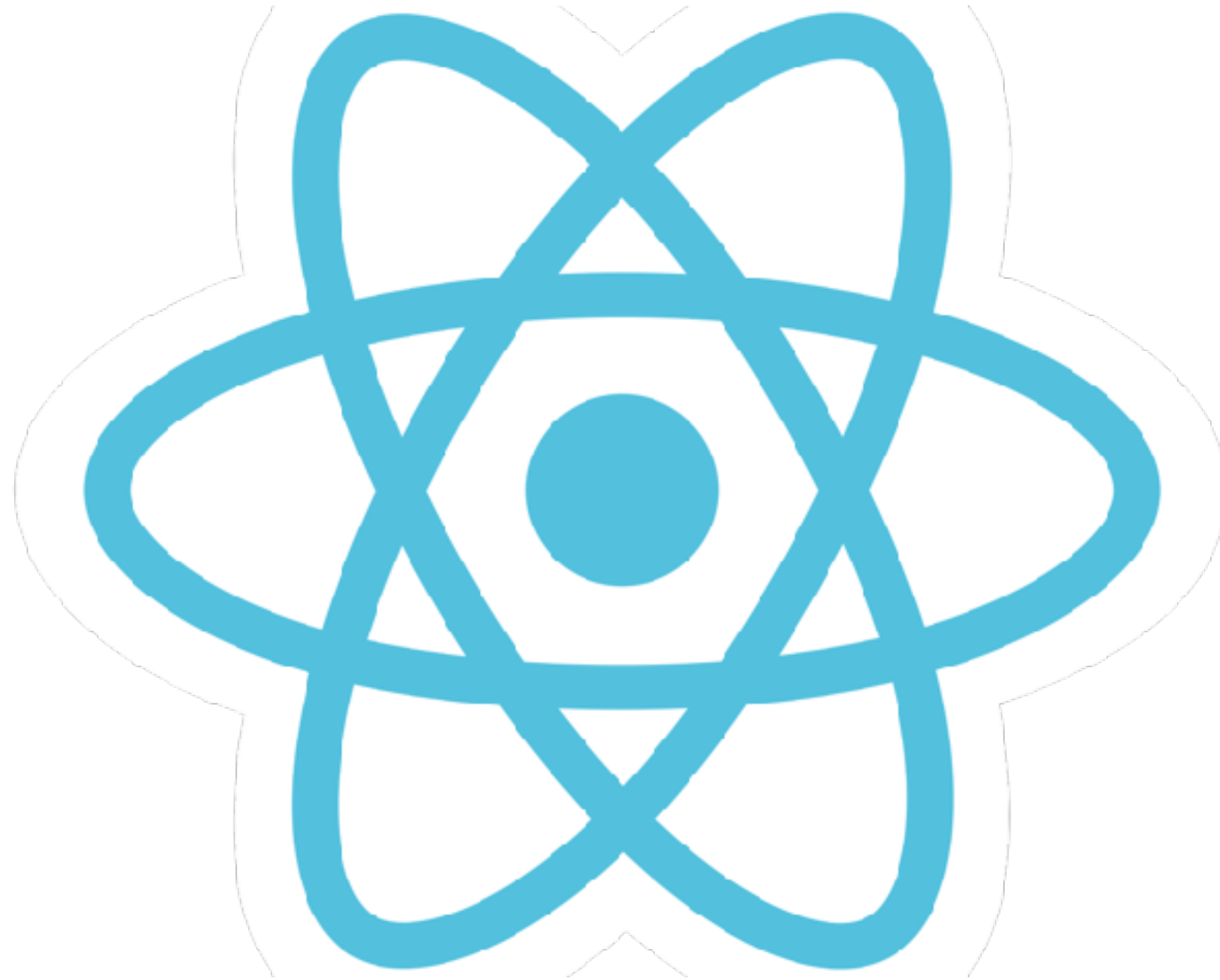- Called after each spec is executed

beforeAll()

- Run a function before any tests in file

- If function returns a promise, will wait for resolve

- can supply a timeout in milliseconds

afterAll

- Run a function after all tests in file complete

- If function returns a promise, will wait for resolve

- can supply a timeout in milliseconds
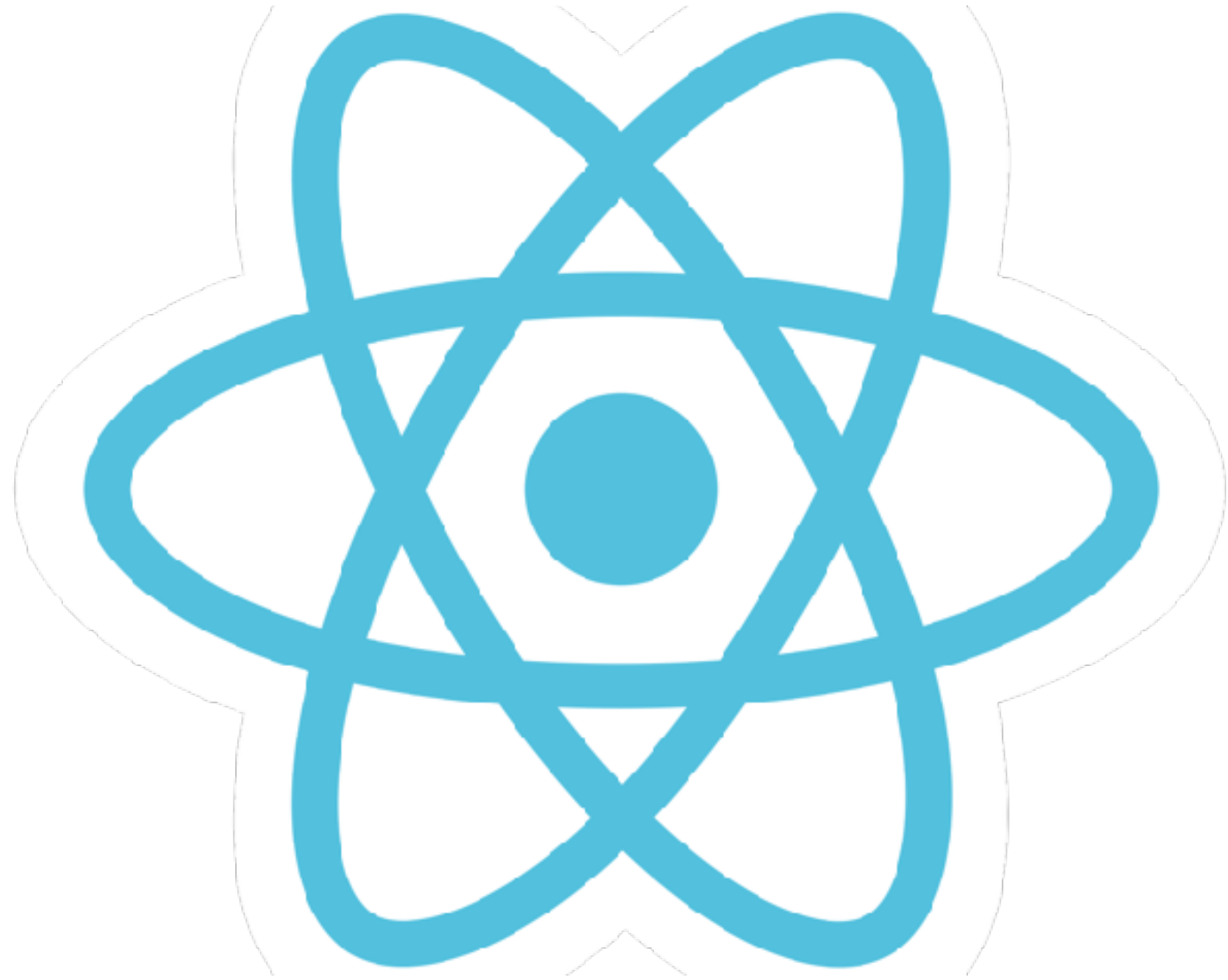
Disabling suites & specs

- prepend an 'x'

- to disable a suite change describe() to xdescribe()

- to disable a spec change it() to xit()

- They are not execute but appear in reporting

Forms

Forms

- Submission

- Validation
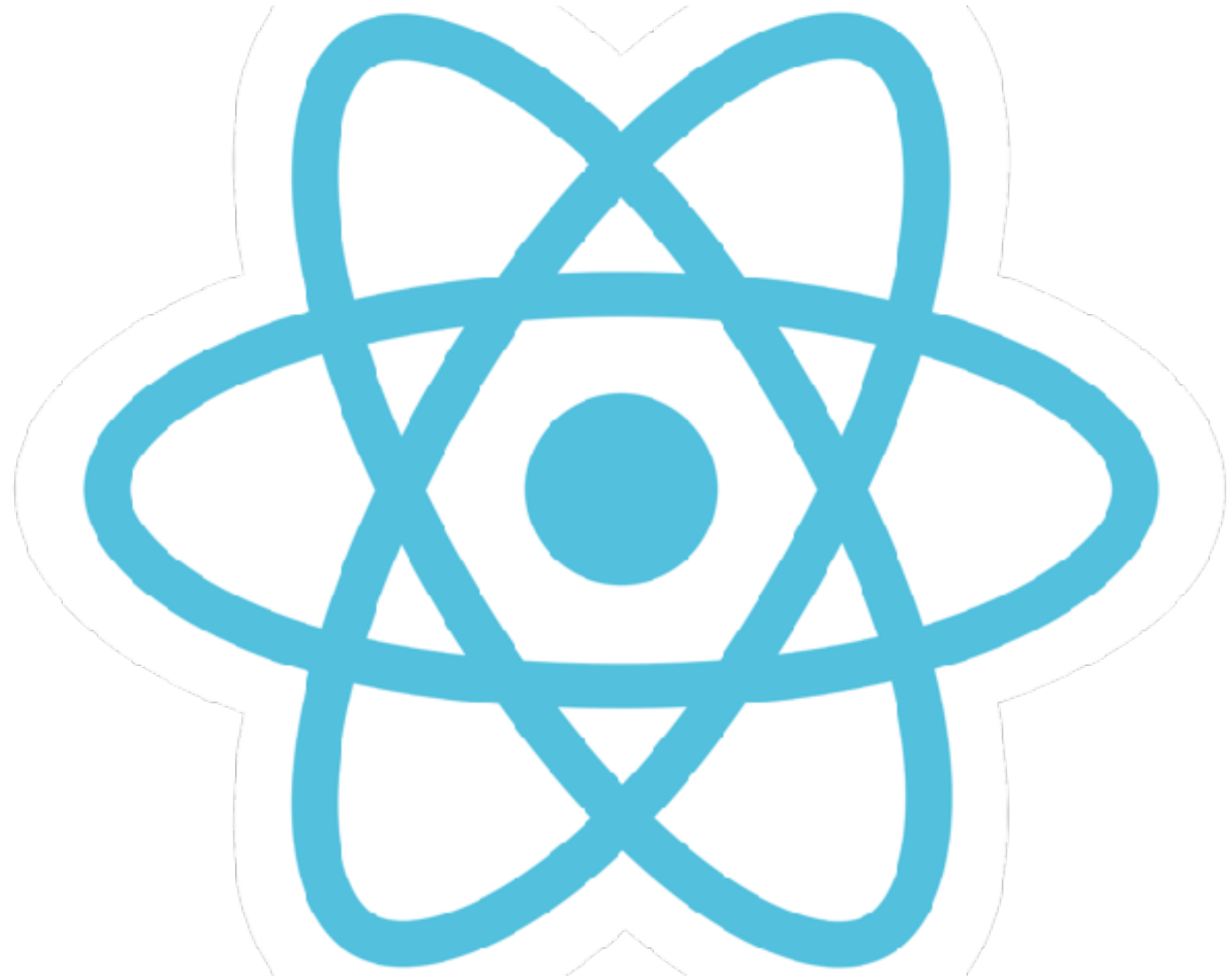
- Hands-on: Forms

Building Apps

Building Apps

- Prop-types

- Custom types

- Composition

- State vs Props

- Data flow

- Component Lifecycle & Methods

Lifecycle Methods

Component	Command
componentWillMount	before initial rendering
componentDidMount	after initial render
componentWillReceiveProps	when new props sent
shouldComponentUpdate	used to override rendering
componentWillUpdate	before rendering when new s/p
componentDidUpdate	after subsequent renders
componentWillUnmount	before component destroyed
componentDidCatch	handles uncaught errors

Debugging

Debugging

- Work top down, outside in

- Begin at parent work down to children

- Replace complex components with simple ones

- Use the debugger and console.log() to validate data

Debugging Tip

- Assign component to a variable

- console.log() the variable

Debugging Tip - JSX

```jsx
const timerComp = (
  <Timer troy='miles'
         { ... this.props}
         onSetClick={this.handleSetClick}
  />
);
console.log(timerComp);
```

Debugging Tip - JSX

Navigated to http://localhost:3000/

timer-dashboard.js:183

▶ Object {$$typeof: Symbol(react.element), key: null, ref: null, props: Object, type: function…}

> |

Navigated to http://localhost:3000/

timer-dashboard.js:183

▼ Object {$$typeof: Symbol(react.element), key: null, ref: null, props: Object, type: function…} ℹ
    $$typeof: Symbol(react.element)
    key: null
  ▼ props: Object
      amount: "15"
      elapsed: 0
      id: 1
    ▶ onResetClick: function (timerId)
    ▶ onSetClick: function ()
    ▶ onStartClick: function (timerId)
    ▶ onStopClick: function (timerId)
      runningSince: undefined
      troy: "miles"
    ▶ __proto__: Object
    ref: null
  ▶ type: function Timer()
  ▶ _owner: ReactCompositeComponentWrapper
  ▶ _store: Object
  ▶ _self: EditableTimer
  ▶ _source: Object
  ▶ __proto__: Object

>

React Router

Routing

- React is a UI library

- It has no built-in routing

- Creating your own solution isn't hard but…

- There are plenty of open source routers

Open Source Routers

- [React Mini Router](#)

- [React Router](#)

- [React Router Component](#)

- [router5](#)

- [Universal Router](#)

React Router

- A complete routing library for React

- Keeps UI in sync with URL

- version 4.2.2

- **yarn add react-router-dom**

Router Basics

- Routing is done with components

- The Router component wraps all

- Route components match paths

- Routes can be nested

Components & Attributes

- <Route/> - matches a path to a component

- <Link/> - creates a link to a Route

- <NavLink/> - a link with styling when matched

- path - defines a route and possible parameters

- component - component called if matched

Route Properties

- props passed to a route

- match - how this route was matched

- history - HTML history

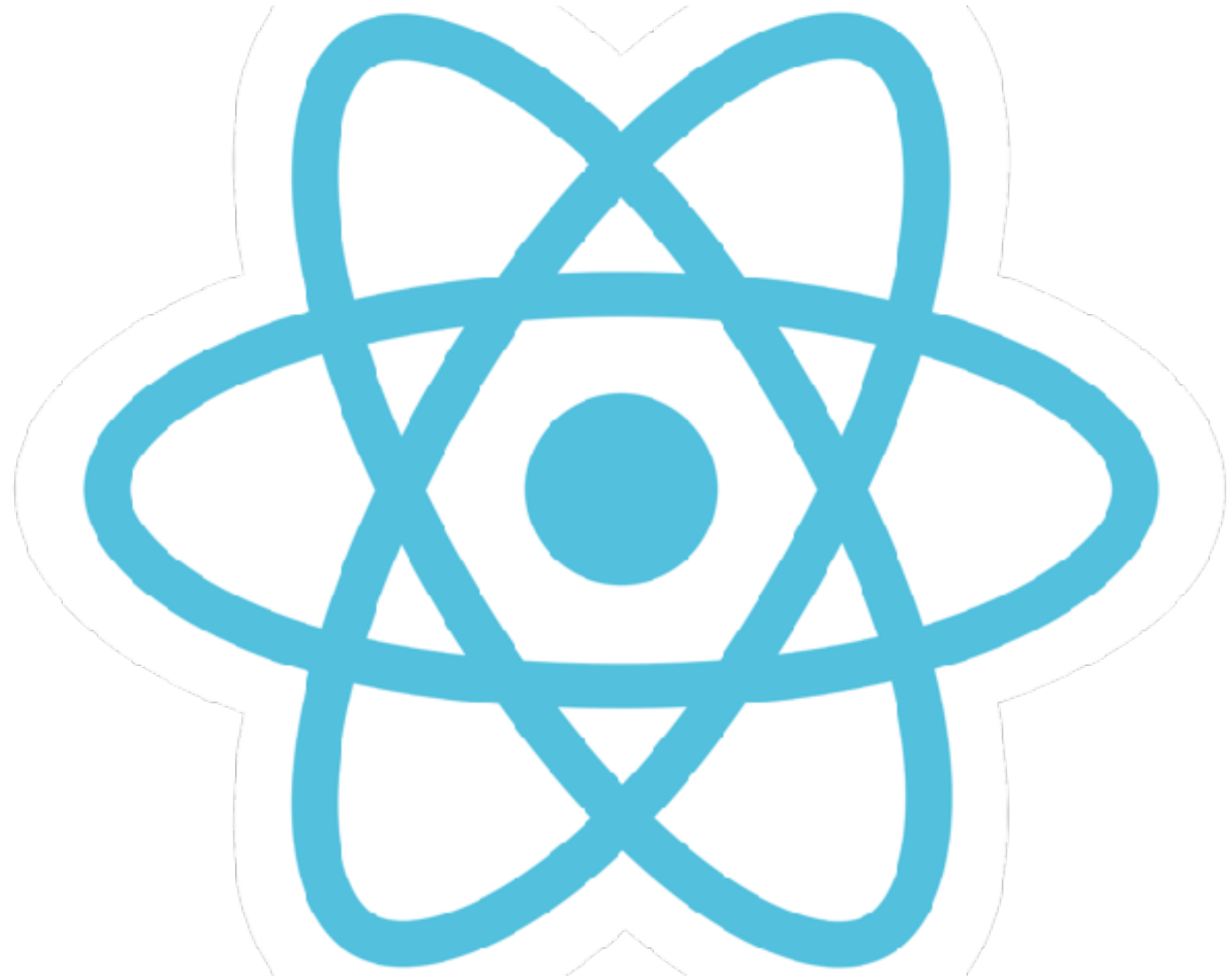- location - where the app is now
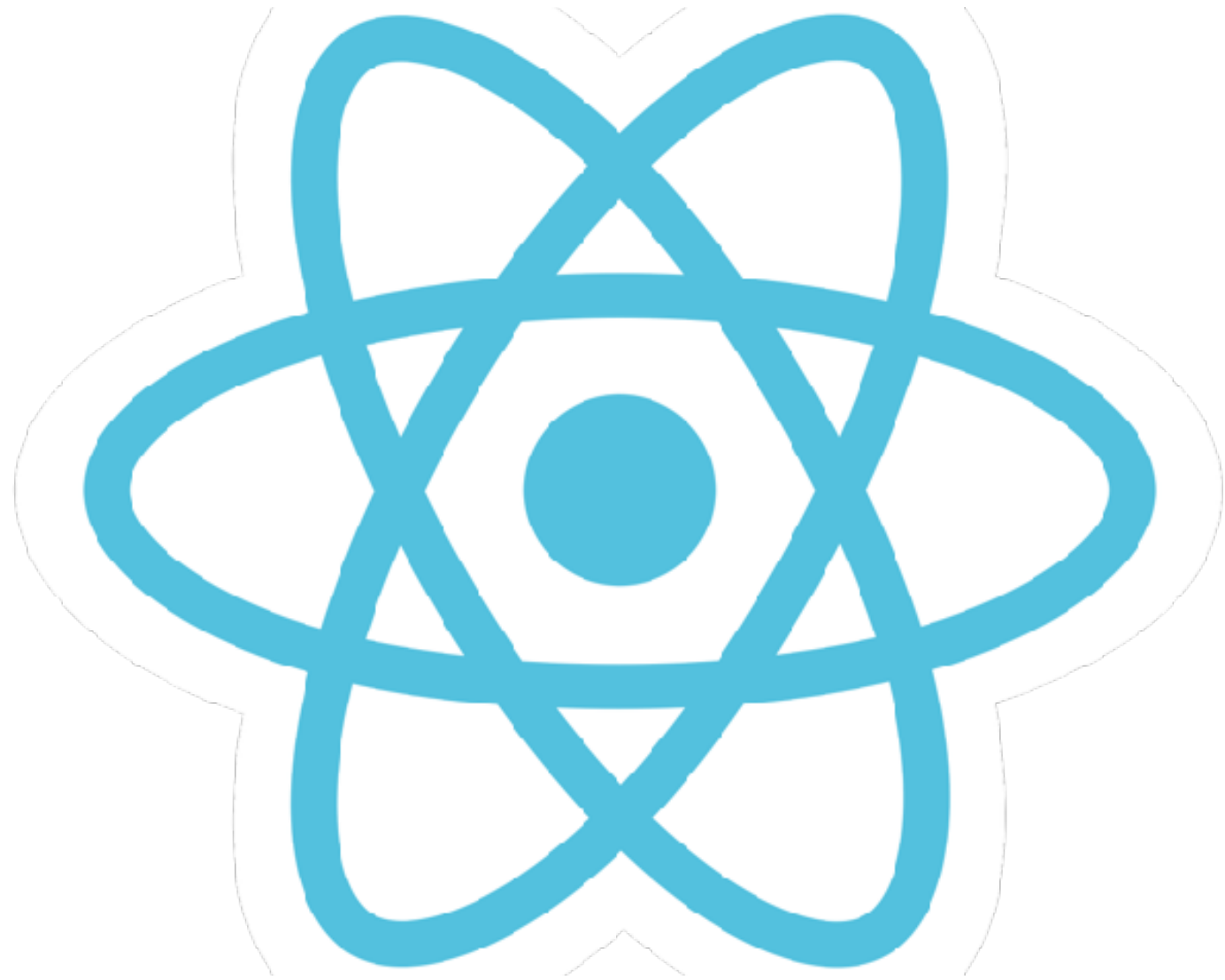
Animation

Animation

- react-addons-css-transition-group

- ReactCSSTransitionGroup

- CSS Keyframe Animation

- Hands-on: Animation

Firebase

Firebase Deploy

1. Create project in Firebase

2. Go to application root in terminal/cmd prompt

3. firebase login

4. firebase init

5. yarn build

6. firebase deploy

React Universal

Server-Side Rendering

- SPA have problems:

    - loading the first page is slow

    - may crash when entered sideways

- Server-side rendering (SSR) solves both issues

- It delivers a fully rendered page initially

Why Server-side Routing?

- Server-side routing helps robots crawl your site*

- Server-side routing usually improves page load speed

- Permits users to enter site on other pages

Why not use SSR?

- It adds a lot of complexity to your site

Implementation

- Requires 2 pieces:

- react-dom/server includes a method renderToString

- react-router-dom includes StaticRouter

Redux

Redux

- A predictable state container for JS apps

- An alternative to and inspired by Flux

- Single store for the entire app

- Makes it easier to hot-load your app

- Created by Dan Abramov

Flux

Flux

- Flux is a design pattern

- Application architecture for building user interfaces

- A pattern for managing data flow in your app

- One way data flow

- 4 main parts: Dispatcher, Store, Action, & View

The 4 main parts

- Dispatcher: receives actions & dispatches them to stores

- Store: holds the data of an app

- Action: define app's internal API

- View: displays the data from stores
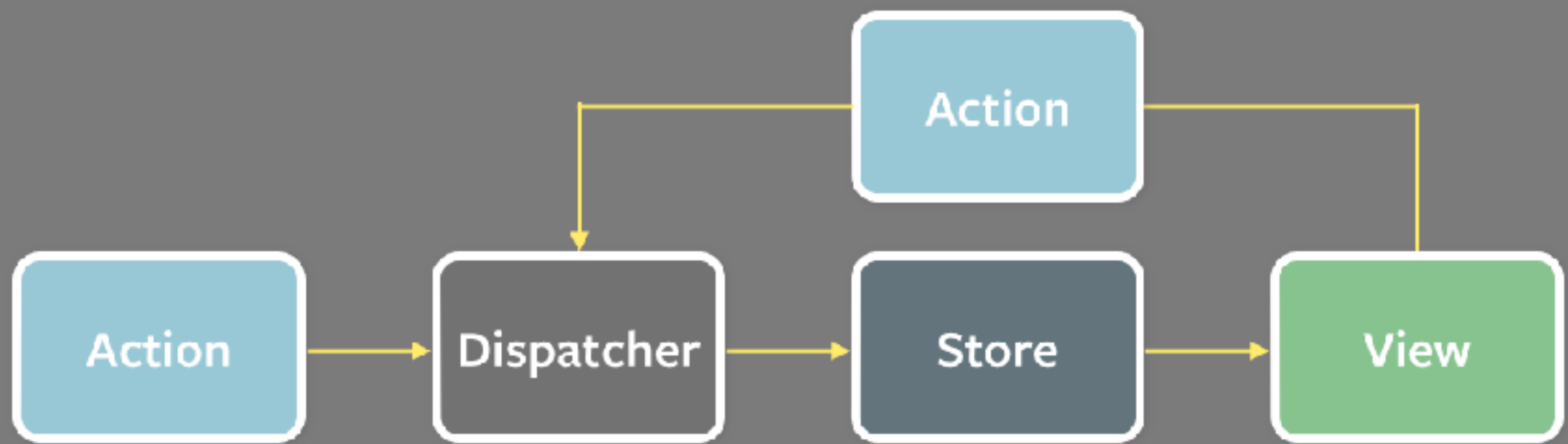
3 Principles

- Single source of truth

- State is read-only

- Changes are made with pure functions

Actions

- Actions are payloads of information that send data from you application to your store.

- They are sent using store.dispatch()

- They are JavaScript objects

- Typically have a type property which defines their action

- The type is normally a string

Action Creators

- Functions that create actions

- This makes them both portable and easy to test

- (In Flux, the creator usually triggers a dispatch)

Reducers

- Take the current state and an action and return the new state

- It is important for reducers to stay pure

A Reducer Shouldn't

- Mutate its arguments

- Perform side effects like API calls or routing transitions

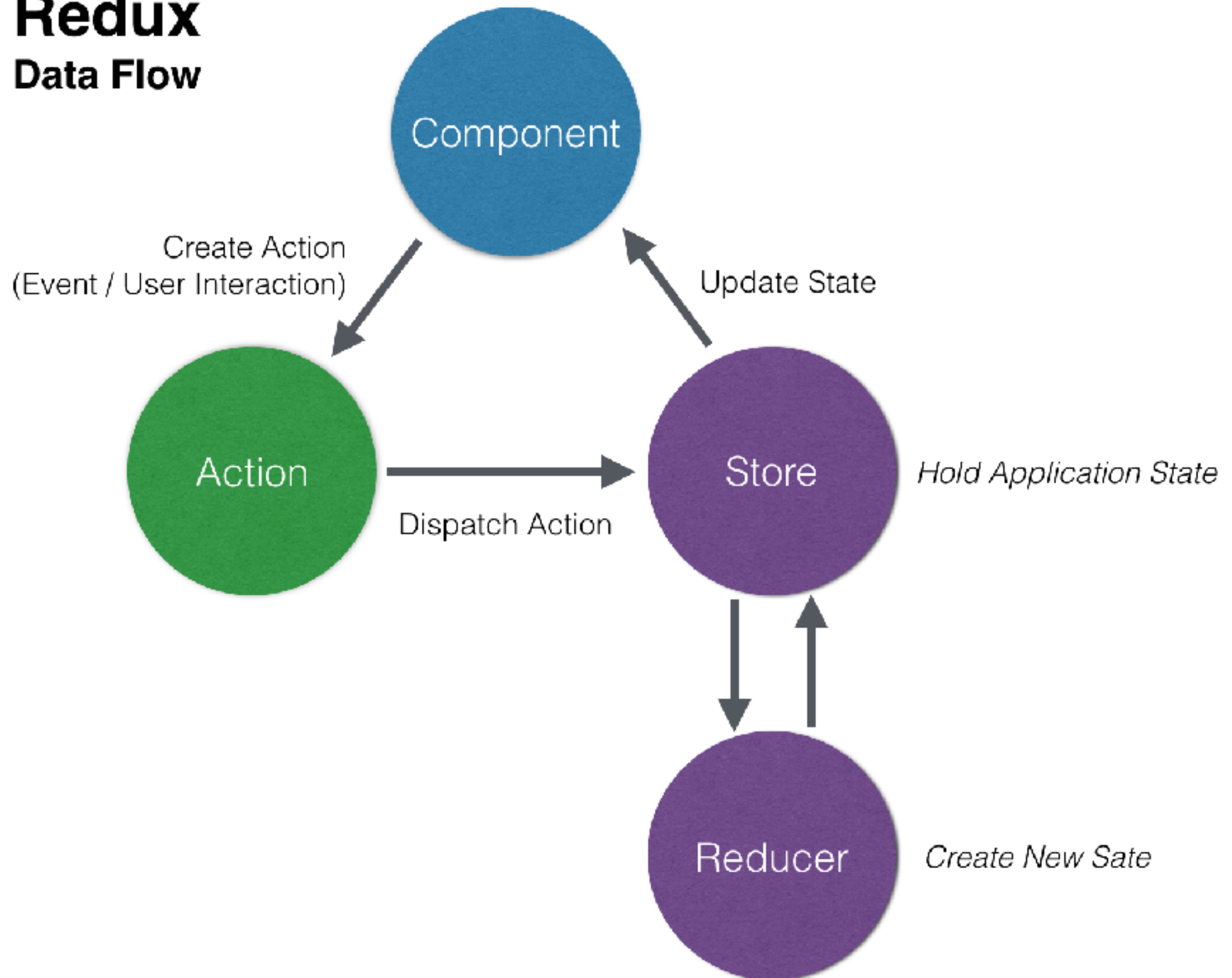- Call non-pure functions like Date.now()

Store

- Holds app state

- Allows access to the state via getState()

- Allows state to be updated via dispatch(action)

- Registers listeners via subscribe(listener)

Redux Data Lifecycle

- Call store.dispatch(action)

- Redux store calls the reducer function

- The root reducer combines the output of multiple reducers into a single state tree

- The redux store saves the complete state tree

- (component.setState(newState) called)

Redux
Data Flow

Component

Create Action
(Event / User Interaction)

Update State

Action

Dispatch Action

Store

Hold Application State

Reducer

Create New Sate

Redux

- A predictable state container for JS apps

- An alternative to and inspired by Flux

- Single store for the entire app

- Makes it easier to hot-load your app

- Created by Dan Abramov

Actions

- Actions are payloads of information that send data from you application to your store.

- They are sent using store.dispatch()

- They are JavaScript objects

- Typically have a type property which defines their action

- The type is normally a string

Action Creators

- Functions that create actions

- This makes them both portable and easy to test

- (In Flux, the creator usually triggers a dispatch)

3 Principles

- Single source of truth

- State is read-only

- Changes are made with pure functions

Reducers

- Take the current state and an action and return the new state

- It is important for reducers to stay pure

A Reducer Shouldn't

- Mutate its arguments

- Perform side effects like API calls or routing transitions

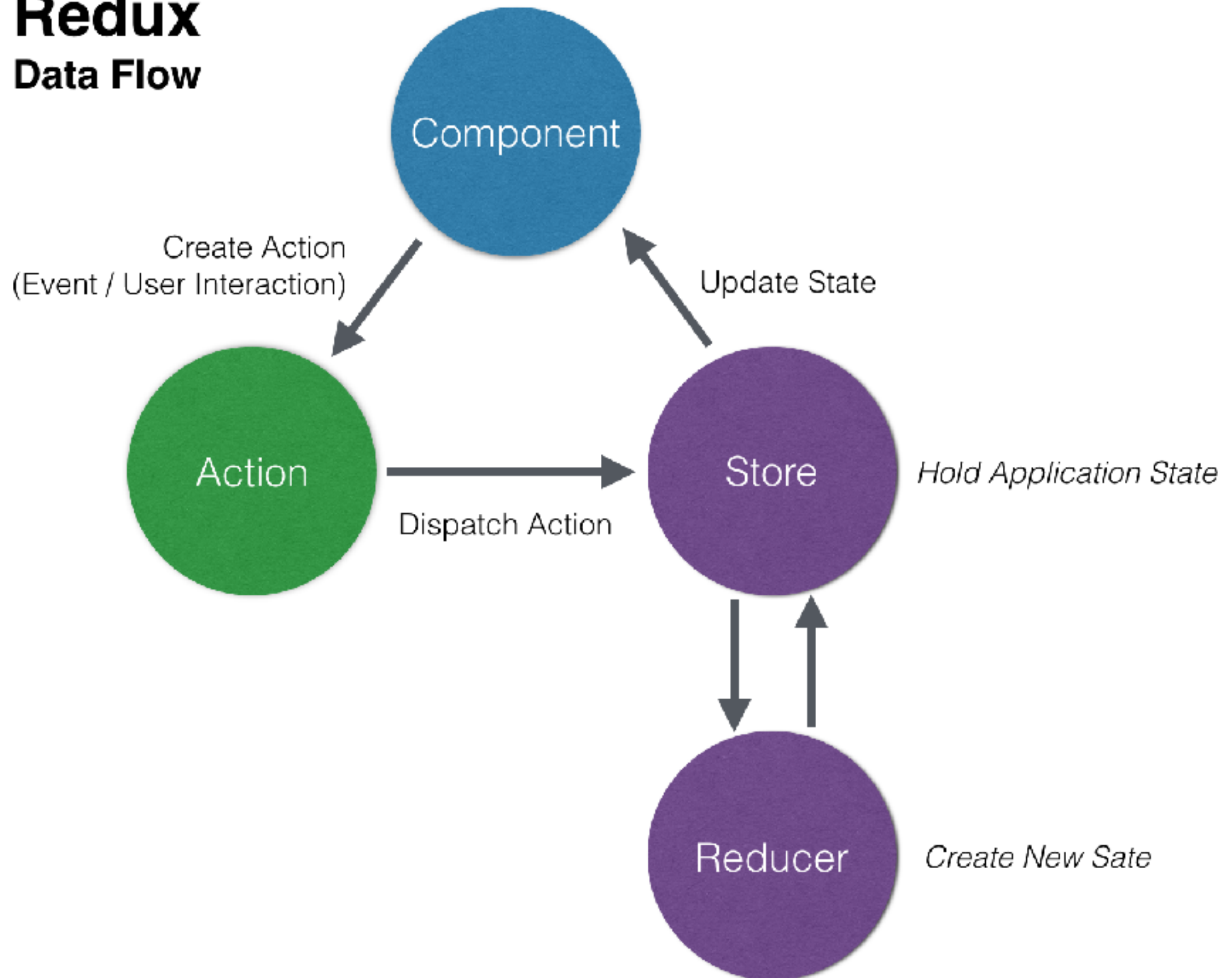- Call non-pure functions like Date.now()

Store

- Holds app state

- Allows access to the state via getState()

- Allows state to be updated via dispatch(action)

- Registers listeners via subscribe(listener)

Redux Data Lifecycle

- Call store.dispatch(action)

- Redux store calls the reducer function

- The root reducer combines the output of multiple reducers into a single state tree

- The redux store saves the complete state tree

- (component.setState(newState) called)

Redux
Data Flow

Component

Create Action
(Event / User Interaction)

Update State

Action

Dispatch Action

Store

Hold Application State

Reducer

Create New Sate

Middleware

- Redux is optimized for a synchronous workflow

- Middleware occurs between the dispatcher and the reducer

- Can be used for: logging, optimistic updating, etc.

React-Redux

- Binds Redux to React

- Redux middleware

- npm i -S react-redux

context

- Similar to props except doesn't have to be explicitly passed

- Any child can have access to context no matter how far down it is

- Libraries use context to create provider components

connect()

- Separates the concerns of the container and presentational components

- Takes 2 arguments

  - a function that maps app state to a JS object

  - a function that maps the store's dispatch to props

- returns a new function

Binders

- mapStateToProps(state)

  - Maps state values to the component's props

- mapDispatchToProps

  - Binds dispatch actions of the component to the store

Component Types

	Presentational	Container
Purpose	How things look	How things work
Redux Aware	No	Yes
Read data	Read from props	Subscribe to Redux State
Change data	Invoke callbacks	Dispatch Redux actions
How written	By hand	Generated by react-redux
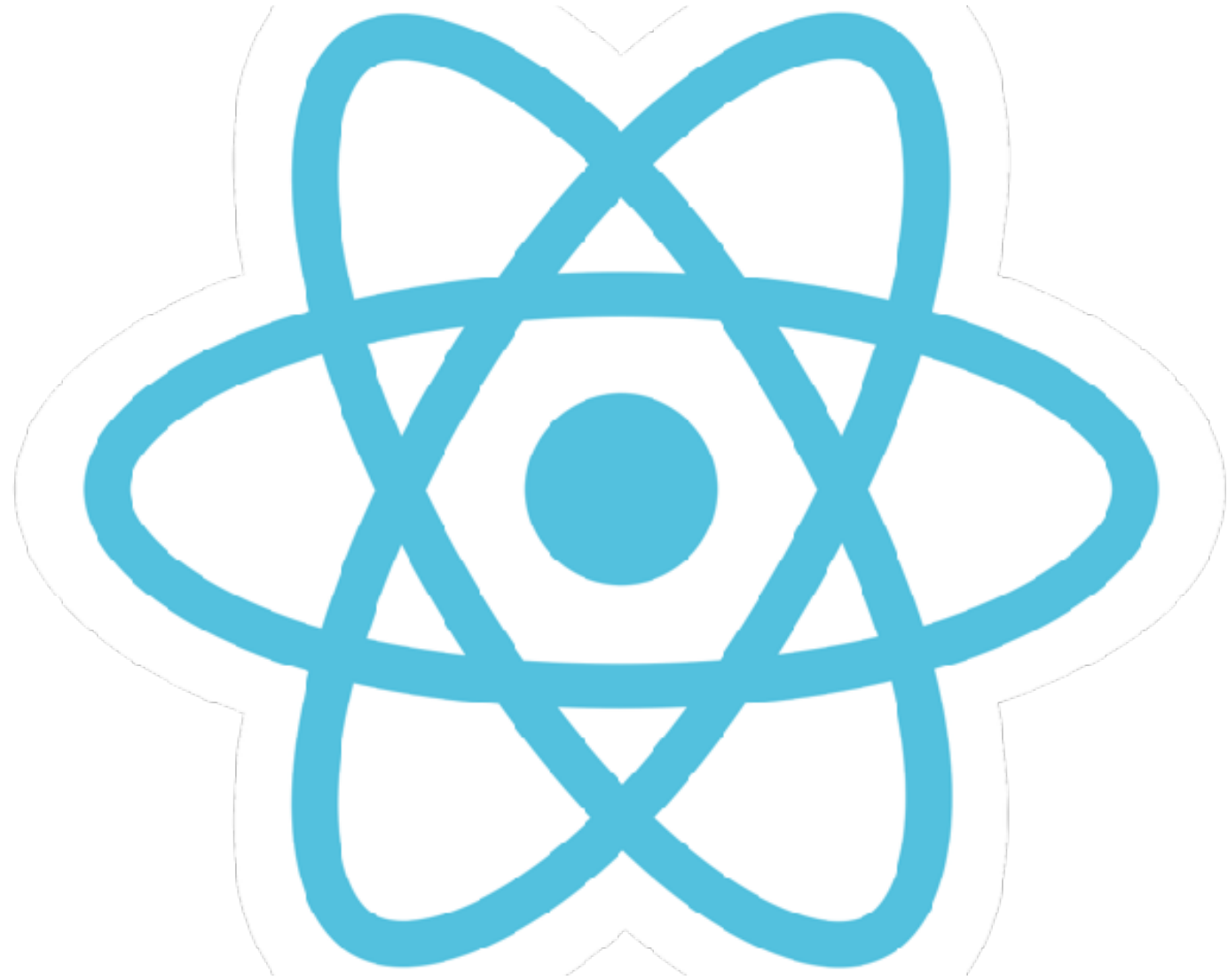
Best Practices

Redux Patterns

Redux Middleware

Redux DevTools

Redux devtools

- Redux devtools is a Chrome extensions

- To activate, add code to your Redux store creation

```
7    const store = createStore(
8      reducer,
9      window.__REDUX_DEVTOOLS_EXTENSION__ &&
10     window.__REDUX_DEVTOOLS_EXTENSION__()
11   );
```
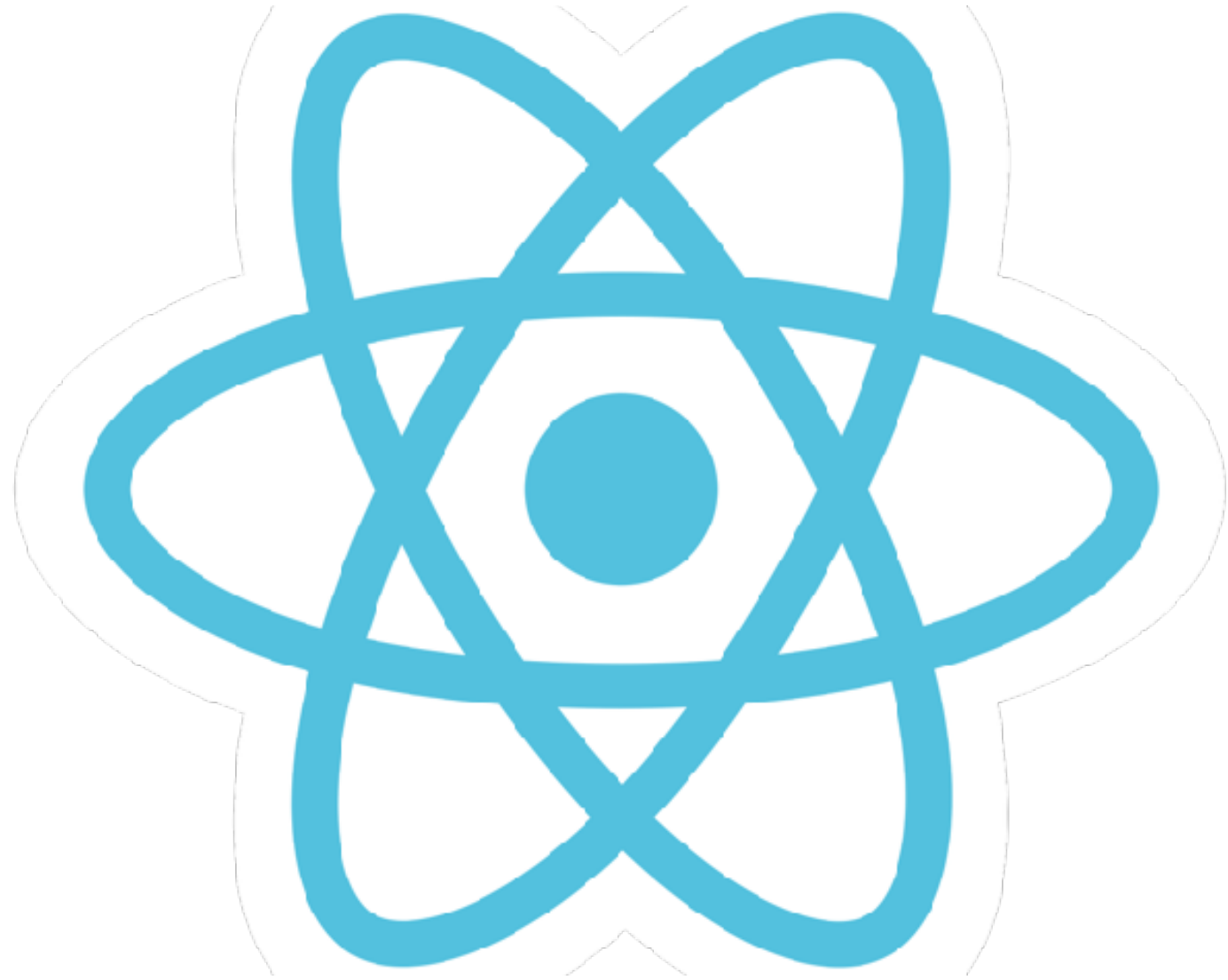
Logger

Redux-Sagas

Saga Helpers/Effects

Testing Sagas

Architecting Apps

Summary

- React is a lightweight library for creating UIs

- Redux makes it better & more predictable

- Redux is easier to use than Flux