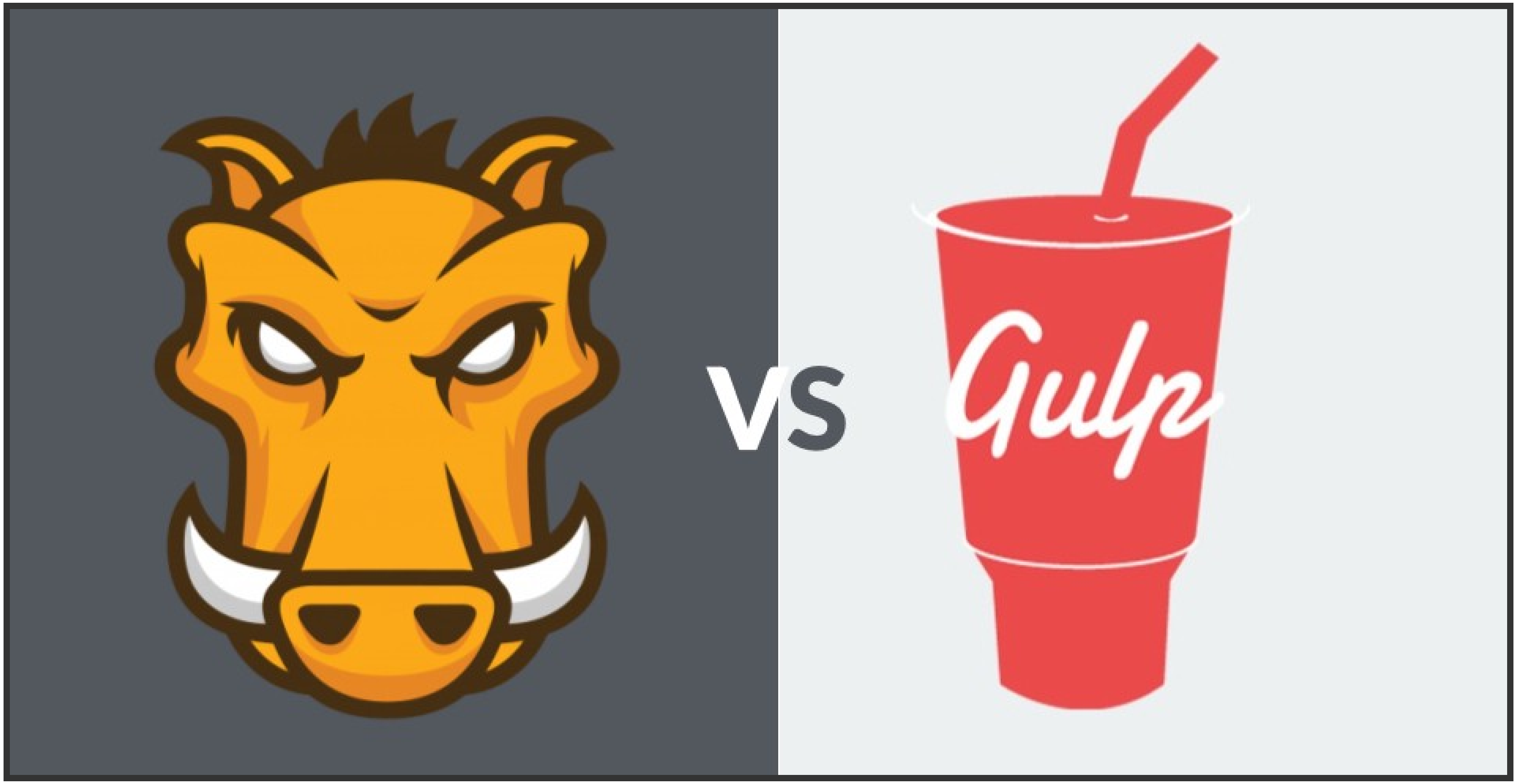
WELCOME TO DAY 0 OF REACTJS BOOTCAMP

AGENDA

- Grunt and Gulp
- Webpack
- Babel
- Eslint
- Arrows
- let
- Destructuring
- Default, Rest, Spread
- Classes
- Module Loader


vs
Gulp

GRUNT

GRUNT

- Focus on configuration
- Does common tasks very well and very easily configured when going down a happy path
- Picks up and drops files from src and dest options so each task opens file readers/writers

```
grunt.initConfig({
clean: {
  src: ['build/app.js', 'build/vendor.js']
},

copy: {
  files: [{
    src: 'build/app.js',
    dest: 'build/dist/app.js'
  }]
}

concat: {
  'build/app.js': ['build/vendors.js', 'build/app.js']
}

//     other task configurations
```

GULP

- Focus on code
- Leverages streams for piping inbetween tasks
- Doesn't enforce much of anything. Just use code to wire up tasks and pipe information

```
//import the necessary gulp plugins
var gulp = require('gulp');
var sass = require('gulp-sass');
var minifyCss = require('gulp-minify-css');
var rename = require('gulp-rename');

//declare the task
gulp.task('sass', function(done) {
  gulp.src('./scss/ionic.app.scss')
    .pipe(sass())
    .pipe(gulp.dest('./www/css/'))
    .pipe(minifyCss({
      keepSpecialComments: 0
    }))
    .pipe(rename({ extname: '.min.css' }))
    .pipe(gulp.dest('./www/css/'))
    .on('end', done);
```

SCRIPT LOADING

- Allows for modular applications
- Allow us to pull in dependencies when we need them
- Can bundle scripts on a per page basis
- AMD Script loading with require was originally browser implementation of CommonJS Transport
- CommonJS and ES6 are the popular formats over AMD

```
var component = require('../component/component'); //amd and commonjs sy
```

BROWSERIFY

Batteries not included solution

- Built to ship Node modules to browsers
- Big plugin environment to add things like watch, factor-bundles, deAMDify etc
- Manages JS only
- Uses transforms to modify code
- provides pre and post bundle callbacks
- Minimal config

```
var outputs = [ // <- Add new bundle names to this list
  'common',
  'contact',
  'help',
  'enrollment',
  'forgot-credentials',
  'index',
  'initialLogin',
  'login',
  'plan-selection',
  'user-registration',
  'producer-services',
  'reset-password'
];


function generateOutputs(options) {
```

WEBPACK

Batteries included solution

- Our solution for this bootcamp
- Built to be a browser solution with nodejs support
- Bundles all your assets and has loaders to make that easier - great for modularity
- Supports all module formats out of the box
- Complex setup with loaders and etc
- Nice hotloading functionality with its built in dev server

```
var pkg  = require('../package.json'),
    path = require('path');

var DEBUG = process.env.NODE_ENV === 'development';
var TEST = process.env.NODE_ENV === 'test';

module.exports = {
  context: path.join(__dirname, '../public'),
  cache: DEBUG,
  debug: DEBUG,
  watch: DEBUG,
  devtool: DEBUG || TEST ? '#inline-source-map' : false,
  target: 'web',
  entry: './scripts/index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(pkg.config.buildDir)
```

BABEL

- Formerly 6to5 but now handles more than es2015
- Transpiles esnext code into something all browsers can use
- Can transform jsx + do hot loading transformations
- Very up to date and community driven
- Used as a pre-build step when writing esnext in the browser environments

ESLINT

- Extendable code linting and style checking
- Every facet is pluggable
- Built on espree parser
- Lints using AST to evaluate patterns unlike some other linters
- Many great community plugins for frameworks like react

```json
{
  "rules": {
      "eqeqeq": 0,
      "curly": 2,
      "quotes": [2, "double"]
  }
}
```

ES2015

Innovation debt is the cost that companies incur when they don't invest in their developers.

- Peter Bell

- Westin Wrzesinski

ARROWS =>

- Inspired by CoffeeScript
- Bind to outer this
- Not newable
- No arguments psuedo array
- Always Anon
- Upgrade to ES5's `.bind(this)` essentially

CODE

```
//Arrows
var evens = numbers.map(num => num % 2 === 0);
nums.map((x) => x * 2);
//or as a statement body
var specialNums = numbers.map(num => {
  return doSomething(num);
});
// Lexical this
var person = {
  _name: "Westin",
  _friends: ["Not Justin", "Doug", "Brendan", "Igor"],
  printFriends() {
    this._friends.forEach(f =>
      console.log(`${this._name} knows ${f}`));
  }
}
```

LET

Allows for block scoping

```
function() {
  if(x) {
    var foo = 3;
  }
  var baz = 1;
  //foo and baz in same scope due to hoisting
}
```

```
function() {
  if(x) {
    let foo = 3; //only inside the conditional
  }
  var baz = 1;
  //foo and baz NOT in same scope as foo is no longer hoisted
}
```

DESTRUCTURING OBJECT

```
var people = [
  {
    name: 'Westin',
    age: 25
  }
];
people.forEach(function({name, age}) //shorthand if key = value
{
  console.log(name + ":" + age)
});

let { first: f, last: l } = {first: 'westin', last: 'w'}; //assign multi

let [x, y] = ['a', 'b']; // x = 'a'; y = 'b'; //extract multiple values a

let {length : len} = 'abc'; // len = 3 nifty trick to call string.length
```

DESTRUCTURING ARRAY

- Fails quietly to undefined

```
var [month, date, year] = [3, 14, 1977];
//swapping
x = 3;
y = 4;
[x, y] = [y, x];
//ignore an index
var [a, ,b] = [1,2,3];
var doWork = function() {
    return [1, 3, 2];
};
let [, x, y, z] = doWork();
```

DEFAULT, REST, SPREAD

DEFAULT PARAMS

```
function f(x, y=12, z=y) {
  // y is 12 if not passed (or passed as undefined)
  return x + y;
}
f(3) == 15;

let [x=3, y] = []; // x = 3; y = undefined nifty use with destructuring
```

REST

- rest parameters are only the ones that haven't been given a separate name, while the arguments object contains all arguments passed to the function
- the arguments object is not a real array, while rest parameters are Array instances, meaning methods like sort, map, forEach or pop can be applied on it directly
- true array unlike the argument psuedo array

```
function multiply(multiplier, ...theArgs) {
  return theArgs.map(function (element) {
    return multiplier * element;
  });
}
```

SPREAD

Expand array params like Func.apply

```
function sum(x,y,z) {
  return x + y + z;
}
total(1, 2, 3);
//before
total.apply(null, [1,2,3]);
//now
total(...[1,2,3]);

let [x,...y] = 'abc'; // x='a'; y=['b', 'c']; //using with destructuring
```

CLASSES

- just some syntactic sugar for prototype
- we will have supers and constructors

```
class TodoModel {

    constructor(storage) {
        this.storage = storage;
    }

    get todo() {
      return this.storage.get();
    }

    set todo(title) {
      //... can override setters of properties to do same as create
      // todo.x = 'xyz'; will call todo.create('xyz');
      this.create(title);
    }

    create(title) {
```

```
class EnhancedTodoModel extends TodoModel {
    constructor(storage) {
        this.storage = storage;
    }
    save(item) {
        alert('Saving a new task');
        super.save(item);
    }
}
```

MODULES

```
import name from "module-name";
import { member } from "module-name";
import {member as alias } from "module-name";
import { member1 , member2 } from "module-name";
import { member1 , member2 as alias2 , [...] } from "module-name";
import name , { member [ , [...] ] } from "module-name";
import "module-name" as name;
//export syntax
Example 1:
export name1, name2, ..., nameN;
Example 2:
export *;
Example 3:
export default function() {...}
```

CAN HAVE BOTH NAMED AND DEFAULT EXPORTS

Default is really just another named export Default are favored however

```
//------ underscore.js ------
export default function (obj) {
    ...
};
export function each(obj, iterator, context) {
    ...
}
export { each as forEach };
//------ main.js ------
import _, { each } from 'underscore';
```

WELCOME TO DAY 1 OF REACTJS BOOTCAMP

AGENDA

- React as a view layer
- JSX
- Mounting component to DOM
- Simple Components
- Starting the application

WHY THE ^@#$% IS THERE ANOTHER JS FRAMEWORK

MVC ISN'T A HOLY GRAIL

- Decent steam behind an non MVC clientside movement

REACT

- Small Modular Components
- Minimal API Surface Area
- Performant with JSDom

- Simply the view layer
- JSX can be strange at first
- Since it is just a view layer you must enforce good patterns

JUST USE WHAT MAKES SENSE AND ENABLES YOU TO BE PRODUCTIVE!

ANGULAR

- Directives are crazy powerful and declarative
- Filters are awesome
- Can be used modularly with some good insight. see Angular UI
- Great for prototyping

- Two binding with Dirty Checks and Digest Cycle
- Technically inperformant but not noticeable 99% of the time
- Code under the head is insanly complex argueable over engineered.
- Actions automagically trigger digest cycles
- Doesn't feel like writing JS as huge API surface area
- Docs still suck

EMBER

- Efficient data binding with accessors
- Less gotchas than Angular
- Computed Properties
- Amazing docs
- Great for prototyping

- Very heavy and opinionated
- Syntax isn't as clean with accessors
- Highest barrier to entry

BACKBONE

- Binding with accessors but requires setup
- 0 assumptions are made
- Extendable and versatile
- Marionette is ♥

- Less for free but it stays out of your way after
- Backbone is actually pretty great for what it does so ...

JUST DO IT!

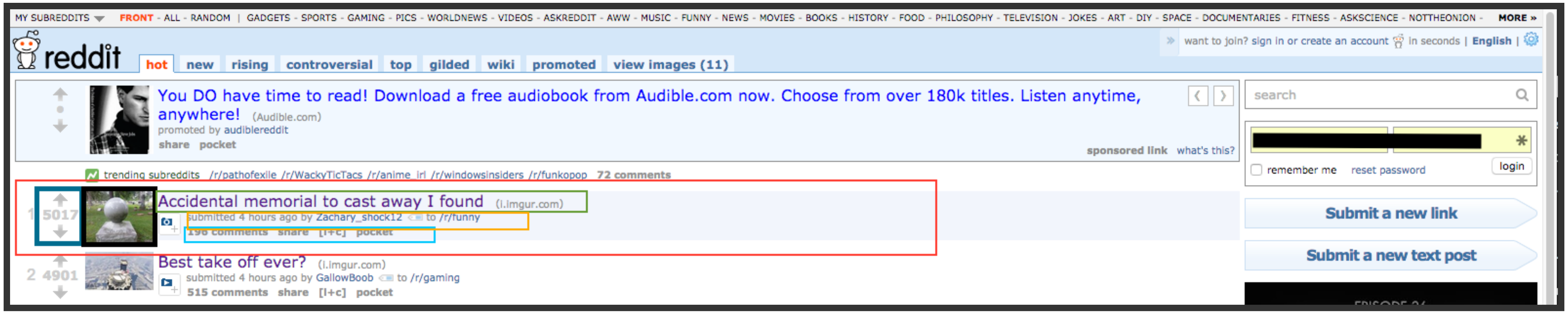
REACT - ACCORDING TO FACEBOOK

- It's simple
- It's declarative
- It's composable

COMPONENTS ARE YOUR BUILDING BLOCKS

- Self contained
- Modular
- Dynamic

HOW CAN WE BREAK THIS DOWN INTO COMPONENTS?


MY SUBREDDITS
FRONT - ALL - RANDOM | GADGETS - SPORTS - GAMING - PICS - WORLDNEWS - VIDEOS - ASKREDDIT - AWW - MUSIC - FUNNY - NEWS - MOVIES - BOOKS - HISTORY - FOOD - PHILOSOPHY - TELEVISION - JOKES - ART - DIY - SPACE - DOCUMENTARIES - FITNESS - ASKSCIENCE - NOTTHEONION - MORE »
reddit
hot new rising controversial top gilded wiki promoted view images (11)
want to join? sign in or create an account in seconds | English |
search
remember me reset password login
Submit a new link
Submit a new text post

You DO have time to read! Download a free audiobook from Audible.com now. Choose from over 180k titles. Listen anytime, anywhere! (Audible.com)
promoted by audiblereddit
share pocket
sponsored link what's this?

[redacted]

trending subreddits /r/pathofexile /r/WackyTicTacs /r/anime_irl /r/windowsinsiders /r/funkopop 72 comments

1 5017 Accidental memorial to cast away I found (i.imgur.com)
submitted 4 hours ago by Zachary_shock12 to /r/funny
196 comments share [l+c] pocket

2 4901 Best take off ever? (i.imgur.com)
submitted 4 hours ago by GallowBoob to /r/gaming
515 comments share [l+c] pocket

Great Comparison Article here

http://derickbailey.com/2015/08/26/building-a-component-based-web-ui-with-modern-javascript-frameworks

JSX

Render markup in your code

- This makes sense with small modular components
- Babel can transpile JSX into plain old javascript
- You can very easily write your javascript logic alongside your presentation layer

MOUNTING A COMPONENT

```
import React from 'react';
import App from './components/App/App';
React.render(
  <app>,
  document.getElementById('app')
);
</app>
```

SIMPLE COMPONENT

```
import React from 'react';
import './_Hai.scss';
export default class Hai extends React.Component {
  render() {
    let img = 'React';
    return (
      <div classname="my-super-awesome-react-app-with-a-class-name">
        <span>Hello Margret, It's me,</span>
        <span>{name}</span>
      </div>
    );
  }
}
```

WELCOME TO DAY 2 OF REACTJS BOOTCAMP

AGENDA

- React Lifecycle Methods
- Composibility
- Passing Props
- Refs

REACT LIFECYCLE METHODS

- render()
- getInitialState()
- getDefaultProps()
- componentWillMount()
- componentDidMount()
- componentWillRecieveProps(nextProps)
- shouldComponentUpdate(nextProps, nextState)
- componentWillUpdate(nextProps, nextState)
- componentDidUpdate(prevProps, prevState)
- componentWillUnmount()

these can not update state

RENDER()

- This is the meat of your component and generates the React Element to draw into the DOM
- Pure function - does not modify state

GETINITIALSTATE()

- return the basic representation of your state. Invocked once before component is mounted

GETDEFAULTPROPS()

- return the basic representation of your props. Invocked once before component is mounted
- values set on this.props in case something does not come in
- Can also set propTypes to enforce type of objects passed in as props
- propType mismatch issues a warning in development

COMPONENTWILLMOUNT()

- Invoked once before initial render
- chance to update state before render

COMPONENTDIDMOUNT()

- Good place to use 3rd party libs or attach listeners or use ajax
- First chance to interact with the DOM node

COMPONENTWILLRECIEVEPROPS()

- invocked when props from parent change
- can react to prop transistion before the rerender is called
- can still access old props via this.props and new props via the first parameter
- calling setState here does not cause an additional render but rather changes state before the render occurs

SHOULDCOMPONENTUPDATE()

- see PureRenderMixin
- can return false to stop a rerender and gain that bit of performance
- defaultly always returns true

COMPONENTWILLUPDATE()

- not called for initial render
- can not call this.setState
- perform prep before rerender

COMPONENTDIDUPDATE()

- not called for initial render
- perform work on domNode

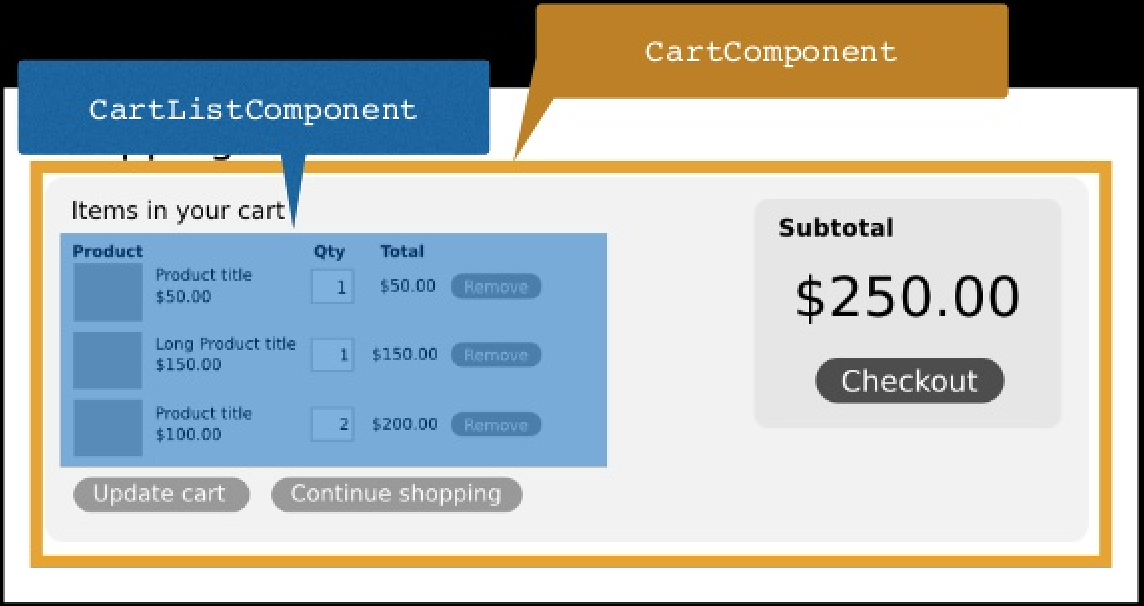
COMPONENTWILLUNMOUNT()

- place to perform teardown code and remove listeners etc

COMPOSIBILITY

- React components can be nested in each other and pass information down to children
- New function syntax for "dumb" components in React 0.14

What if we “separate” another way?



- Notice how Cart Component is a parent to CartListComponent
- CartListComponent may contain child components to render each list item
- With this scheme the list items can own their state of qty and total while the cart list is only concerned with having x amount of generic items

- An owner is the component that sets the props of other components.
- React components can be created and rerendered with props from parent components
- Children usually rendered in the order they appear in the DOM but can dynamically be added like search results

PASSING PROPS

```
<div classname="wrapper">
  <childcomponent proptopass="{this.propToPass}">
</childcomponent></div>
```

REFS

- Reference to a component in that view
- Can be accessed in parent component by this.refs.refName

```
<div classname="wrapper">
  <childcomponent ref="childComponent">
</childcomponent></div>
```

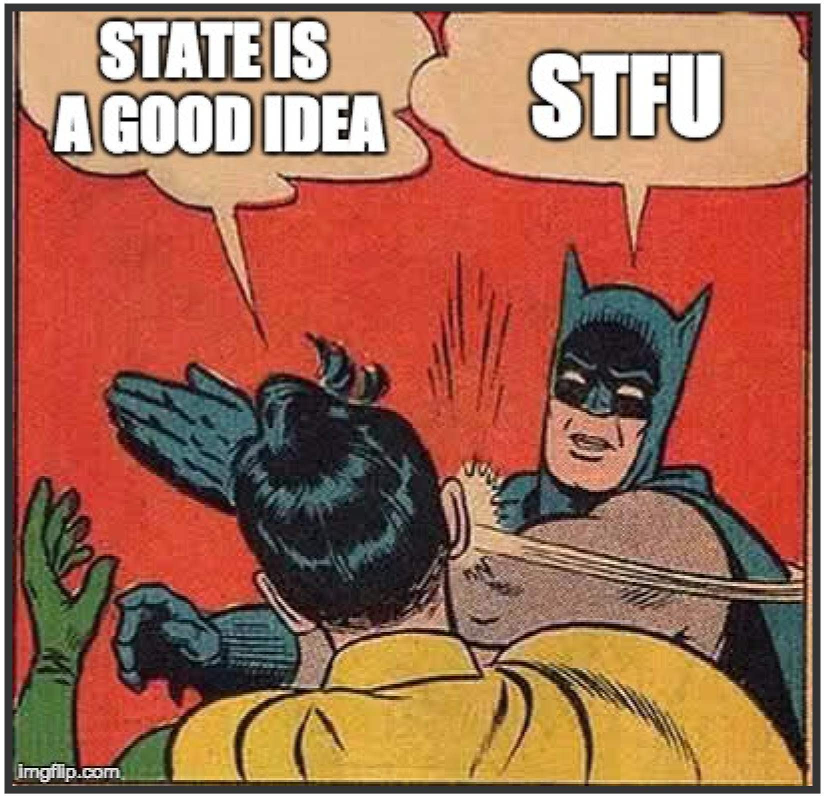
- Can reach out to components public facing methods
- Can reach out to native components like an input tag
- NEVER access a ref during render
- Automagically created and destroyed for you

WELCOME TO DAY 3 OF REACTJS BOOTCAMP

AGENDA

- Component State
- Manipulating States
- Methods of storing data for components

THEY SAID STORE THE SHARED DATA
I USED STATE
imgflip.com


STATE IS A GOOD IDEA
STFU
imgflip.com

@HOLBT

BRI@NHO.LT

... BUT SERIOUSLY

- State isn't that awful but you should be mindful of it while designing your components
- Favor "dumb" stateless components getting data from "smart" parent components

COMPONENT STATE

- Keep state as simple as possible including keeping simple data types
- Only place something on state if the component 100% owns it
- Leave complex calculations on render if possible
- this.state is immutable so use `this.setState({})`
- Less on state means easier testing

MANIPULATING STATE

- Set state is an async function
- Props aren't available during getInitialState
- Can not manipulate state during render lifecycle
- State changes trigger a rerender -> think state changes with dom events
- shouldComponentUpdate can be used for "pure" rendering

SHOULDCOMPONENTUPDATE

```
boolean shouldComponentUpdate(object nextProps, object nextState)
```

- React has a PureRenderMixin but we can easily build one ourselves
- called before component rerender and if it returns false will cancel render
- can use this oppurtunity to check new state and props against old ones
- default returns true
- great to modify for performance

STORING DATA ON COMPONENTS

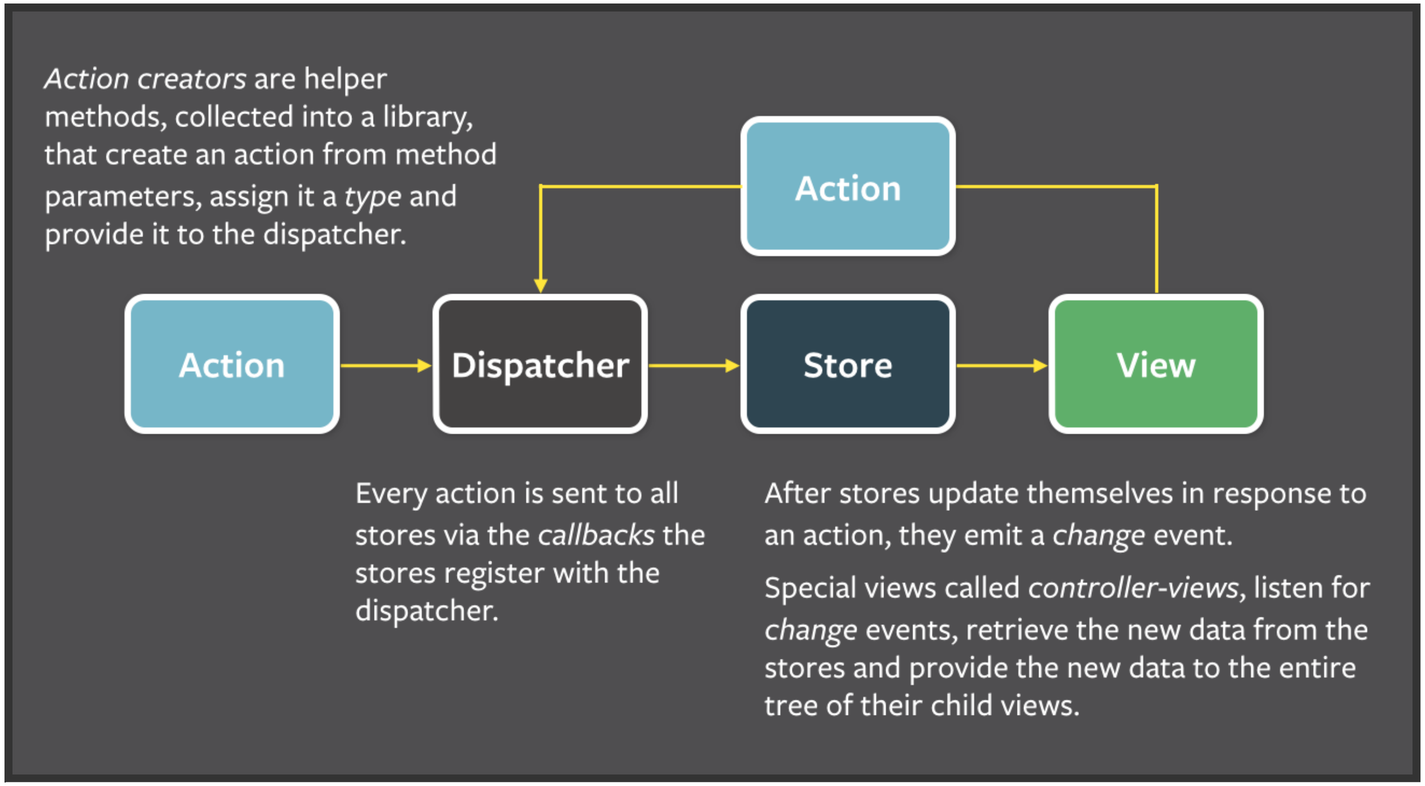
- If we think of a component that needs an ajax call to retrive it's OWN data..
- This is a good time to use state. On xhr completion we can update state and let the component rerender

WELCOME TO DAY 5 OF REACTJS BOOTCAMP

AGENDA

- Introduction to Flux
- Actions, Dispatchers, Stores
- Overall Result of Flux Architecture

FLUX



Action creators are helper methods, collected into a library, that create an action from method parameters, assign it a *type* and provide it to the dispatcher.

Every action is sent to all stores via the *callbacks* the stores register with the dispatcher.

After stores update themselves in response to an action, they emit a *change* event.

Special views called *controller-views*, listen for *change* events, retrieve the new data from the stores and provide the new data to the entire tree of their child views.

Flux is really just a fancy term for pub/sub architecture, i.e. data always flows one way through the application and it is picked up along the way by various subscribers (stores) who are listening to it.

Flux eschews MVC in favor of unidirectional data flow. What this means is that data enters through a single place (your actions) and then flow outward through their state manager (the store) and finally onto the view. The view can then restart the flow by calling other actions in response to user input.

WHY FLUX

- State is messy
- Unidirectional data makes for easy debugging
- Composable components favor reuse
- Stores as a single domain also simplifies debugging

DISPATCHER

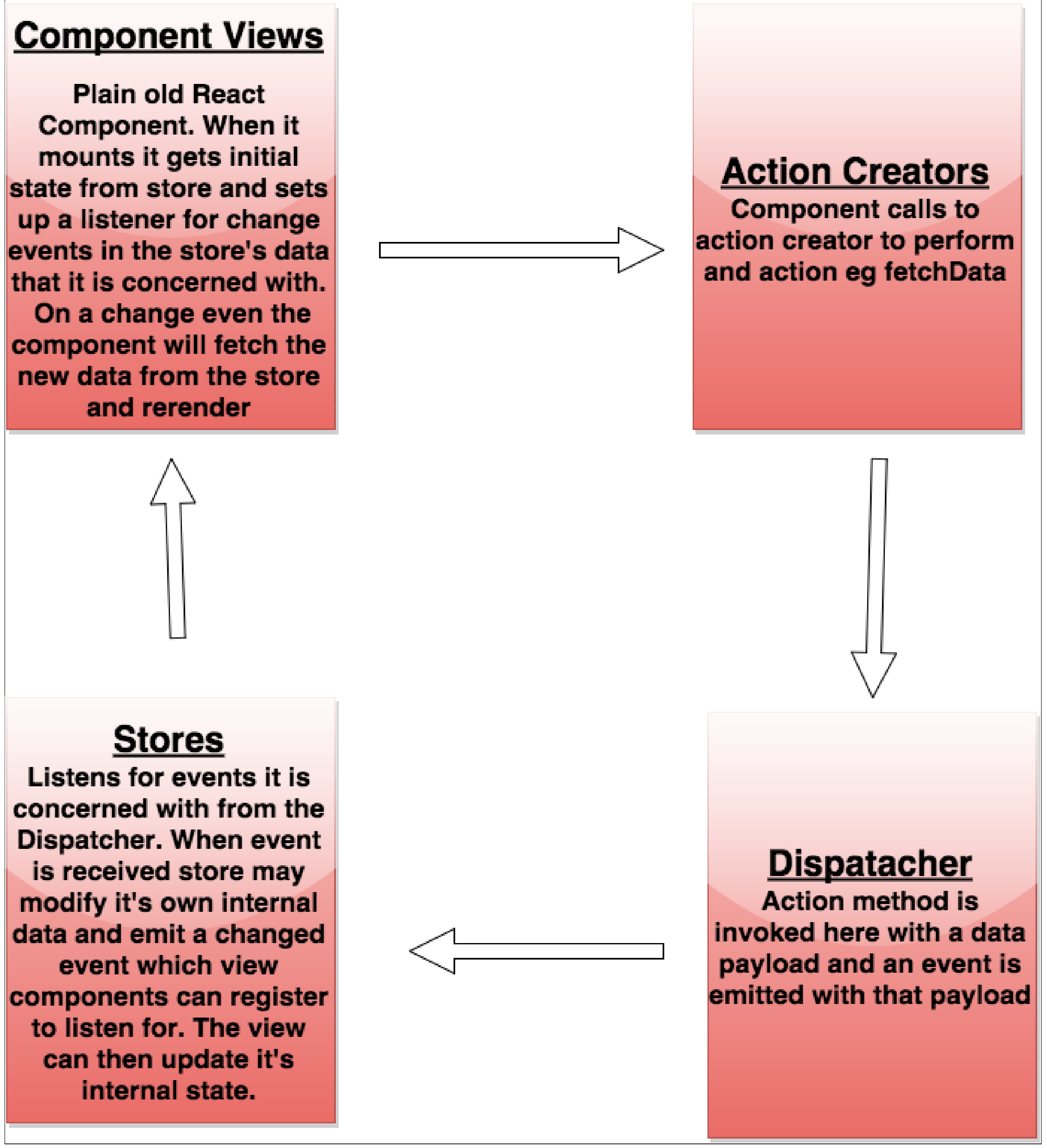
- *SINGLE* messaging hub in the application
- Registry of callbacks
- Has no logic
- Dispatcher recieves actions and fires corresponding callback
- Can manage dependencies between stores

STORES

- Contain the domain logic
- More than just a model
- All stores get the callback from the dispatcher and handle in a case statement
- Changes made through dispatcher -> Store's case statement on action type -> Data update -> Fire change event
- Views can query the store but they are treated as immutable from view's perspective

ACTIONS

- Views can start the communication intents via actions
- Actions are sent through the dispatcher and sent out to the stores
- We have had luck with past tense but it doesn't matter as long as you are consistent
- Try to keep actions generic where possible


Component Views
Plain old React Component. When it mounts it gets initial state from store and sets up a listener for change events in the store's data that it is concerned with. On a change even the component will fetch the new data from the store and rerender
Action Creators
Component calls to action creator to perform and action eg fetchData
Dispatacher
Action method is invoked here with a data payload and an event is emitted with that payload
Stores
Listens for events it is concerned with from the Dispatcher. When event is received store may modify it's own internal data and emit a changed event which view components can register to listen for. The view can then update it's internal state.

WELCOME TO DAY 6 OF REACTJS BOOTCAMP

AGENDA

- Create a router for our application

STORES ARE MORE THAN JUST MODELS

- We can use stores to hold onto state
- This makes stores powerful in routing
- Can update and emit change event to parent views to rerender children
- This demo chooses to go a slightly different route and uses page.js

HIGHER-ORDER FUNCTIONS

- Mixin API for React while stable does not play with es6 classes
- High Order Functions are a more ideal solution for numerous reasons

COMPARE AND CONTRAST THESE SOLUTIONS

```
var array = [1, 2, 3];
for (var i = 0; i < array.length; i++) {
  var current = array[i];
  console.log(current);
}

//vs........

function forEach(array, action) {
  for (var i = 0; i < array.length; i++)
    action(array[i]);
}

forEach([1, 2, 3], console.log);
```

HIGHER-ORDER FUNCTIONS

- Functions that operate on other functions, either by taking them as arguments or by returning them, are called higher-order functions.
- Higher-order functions allow us to abstract over actions, not just values.

Video Here

https://www.youtube.com/watch?v=wfMtDGfHWpA