Angular vs. React Framework Deathmatch

A Software Presentation From

Jeremy Zerr

Site: http://www.zerrtech.com

LinkedIn: http://www.linkedin.com/in/jrzerr

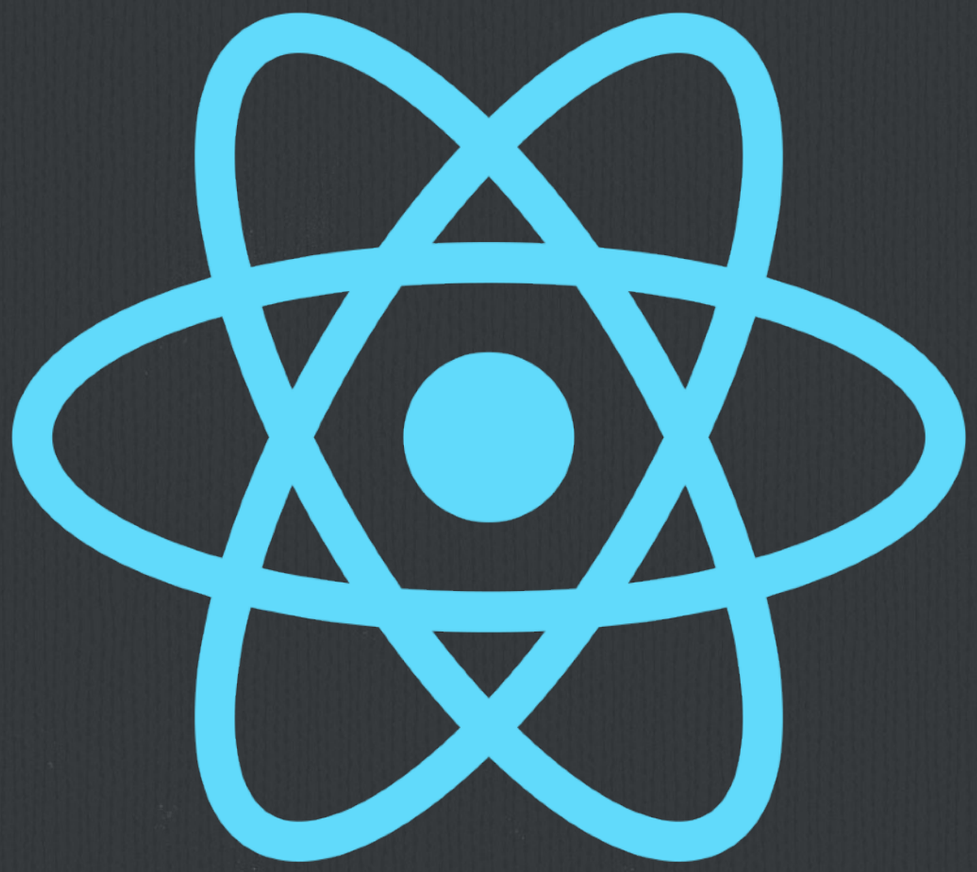
Twitter: http://www.twitter.com/jrzerr

Arguments to settle

- [ ] Tabs vs. Spaces
- [ ] Vi vs. Emacs
- [ ] Angular vs. React

Tabs vs. Spaces

Angular vs. React



What I Do

- [ ] I build custom software
- [ ] We create web applications, mobile apps, front to back
- [ ] Started the business 9 years ago, currently a team of 3

ZERRTECH

We have built Angular or React apps for

Micron

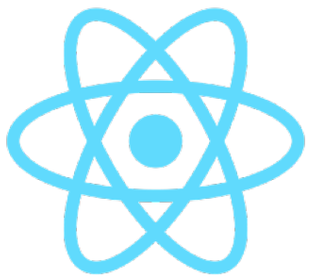
TSHEETS
We ♥ Employees

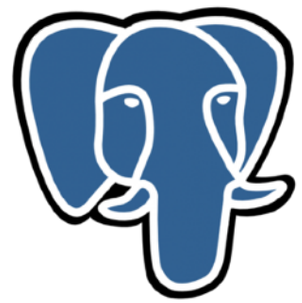
BOISE STATE UNIVERSITY

Kount

Standlee
PREMIUM WESTERN FORAGE®

We also do a lot of other stuff





Weigh In: Angular

- [ ] Currently Angular version 5 (really 2 Angulars, Angular 1 and Angular 2+)
- [ ] Sponsored by Google
- [ ] Favorite of large enterprises due to more all-in-one opinionated solution
- [ ] We have built a couple Angular 2+ apps, and lost count of Angular 1 apps (easily > 5) and including Ionic

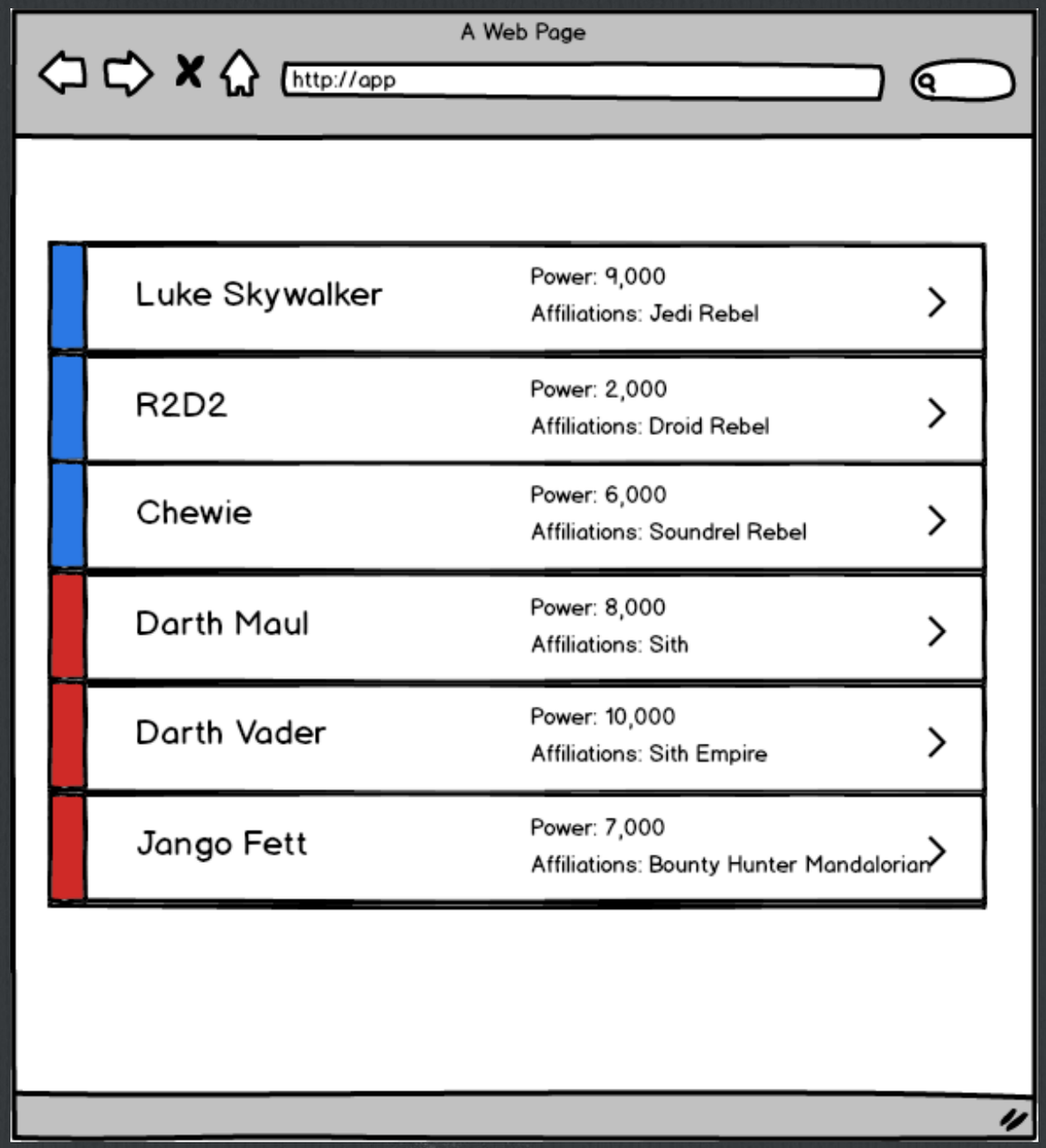
Weigh In: React

- [ ] Currently React version 16.2
- [ ] Sponsored by Facebook
- [ ] Favorite of startups due to working with lots of existing JS libraries to create a site
- [ ] We have built > 7 or so React apps, some React Native

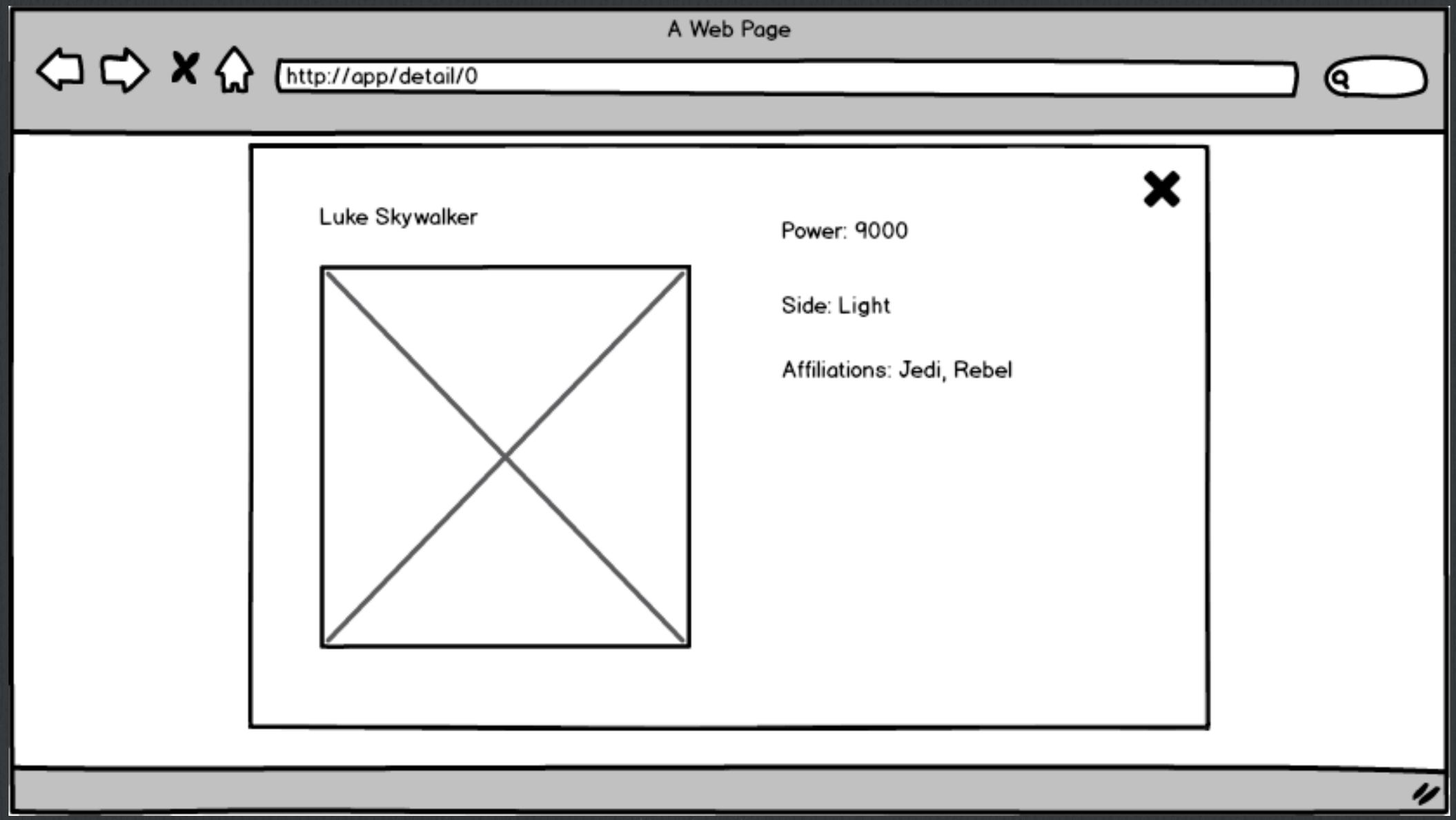
How do we evaluate this?

- ☐ Build a real app with both
- ☐ Add in the types of things big projects need and use them in both
- ☐ Compare code
- ☐ Compare the little things and the big things

Here is what we want to build



Here is what we want to build



Our next design step: What do we store?

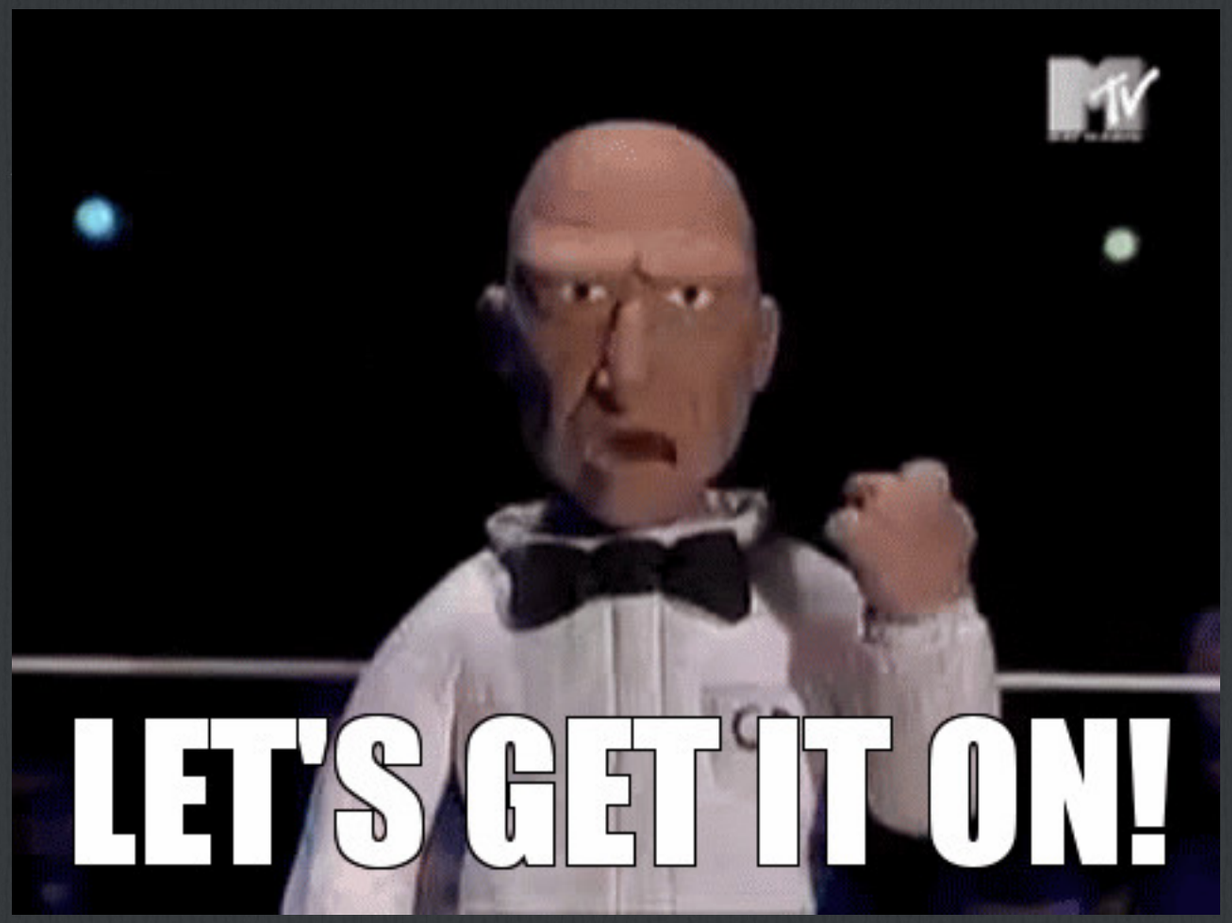
- ☐ We have an API set up on Heroku that returns heroes with Github code if interested
- ☐ We will store:
  - ☐ List of heroes
  - ☐ Loading status of API call
  - ☐ Any error status of API call
- ☐ We will just use two routes to view the selected hero, so we don't really need to store the "active" hero. We'll just use the route as the "store" for that

Onto the tech stack

- ☐ TypeScript
- ☐ Redux for our store w/Dev Tools
- ☐ redux-observable to use Epics for API calls (side effects)
- ☐ Router that integrates with Redux
- ☐ Flux Standard Action
- ☐ Webpack 2
- ☐ Use cmd line tools to start the project

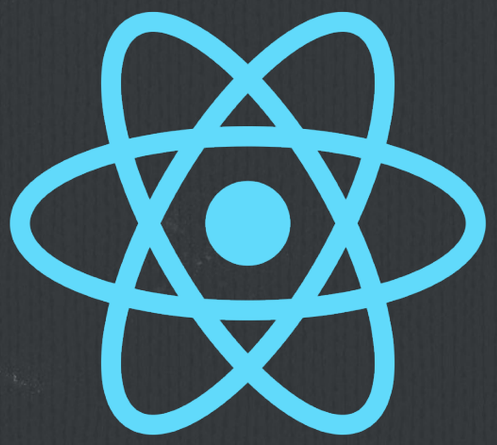
Architecture

- [ ] We don’t want to have to eject from the cmd line tools
- [ ] We want to keep the same file structure
- [ ] We plan to use the container component/dumb component design structure


LET'S GET IT ON!

Github Links

Zerrtech/angular-deathmatch



Zerrtech/react-deathmatch

CLI action

- [ ] We can use the Angular CLI
- [ ] nvm use v6
- [ ] npm install -g @angular/cli
- [ ] ng new angular-deathmatch
- [ ] cd angular-deathmatch
- [ ] rm -rf node_modules
- [ ] yarn install
- [ ] ng serve

- [ ] We can use create-react-app
- [ ] nvm use v6
- [ ] npm install -g create-react-app
- [ ] create-react-app react-deathmatch --scripts-version=react-scripts-ts
- [ ] cd react-deathmatch
- [ ]
- [ ]
- [ ] yarn start

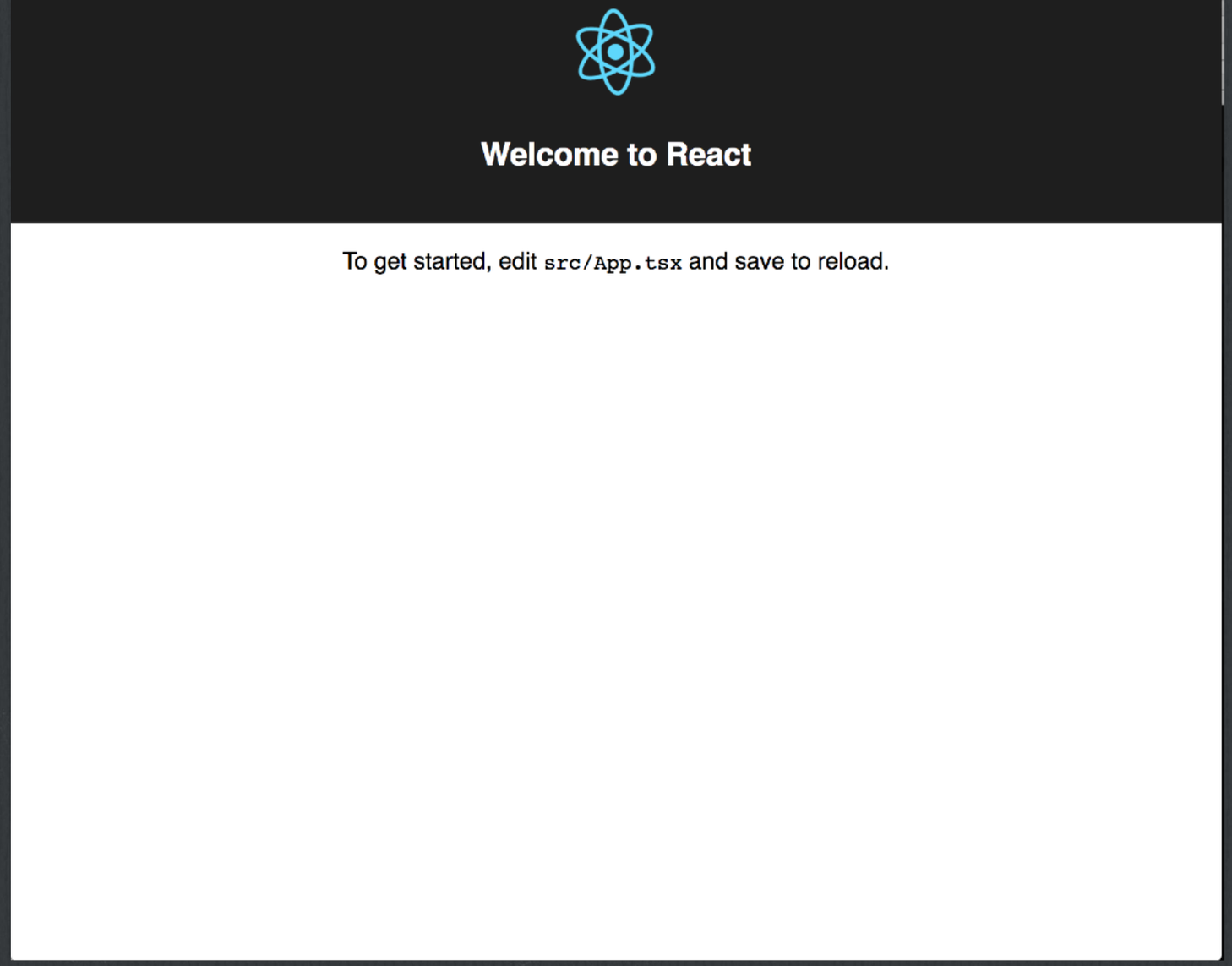
Angular Initial App

Welcome to app!

Here are some links to help you start:

- Tour of Heroes
- CLI Documentation
- Angular blog

React Initial App



Project Structure

REACT-DEATHMATCH

- node_modules
- public
  - favicon.ico
  - index.html
  - manifest.json
- src
  - App.css
  - App.test.tsx
  - App.tsx
  - index.css
  - index.tsx
  - logo.svg
  - registerServiceWorker.ts
- .gitignore
- package.json
- README.md
- tsconfig.json
- tsconfig.test.json
- tslint.json
- yarn.lock

ANGULAR-DEATHMATCH

- e2e
- node_modules
- src
  - app
  - assets
  - environments
  - favicon.ico
  - index.html
  - main.ts
  - polyfills.ts
  - styles.css
  - test.ts
  - tsconfig.app.json
  - tsconfig.spec.json
  - typings.d.ts
- .angular-cli.json
- .editorconfig
- .gitignore
- karma.conf.js
- package.json
- protractor.conf.js
- README.md
- tsconfig.json
- tslint.json
- yarn.lock

Let's install the same libs

- [ ] yarn add redux
- [ ] yarn add rxjs (Observe all the things!)
- [ ] yarn add redux-observable (Epics!)
- [ ] yarn add redux-logger
- [ ] yarn add flux-standard-action (More strict actions!)

Wha!

Fast Foward...

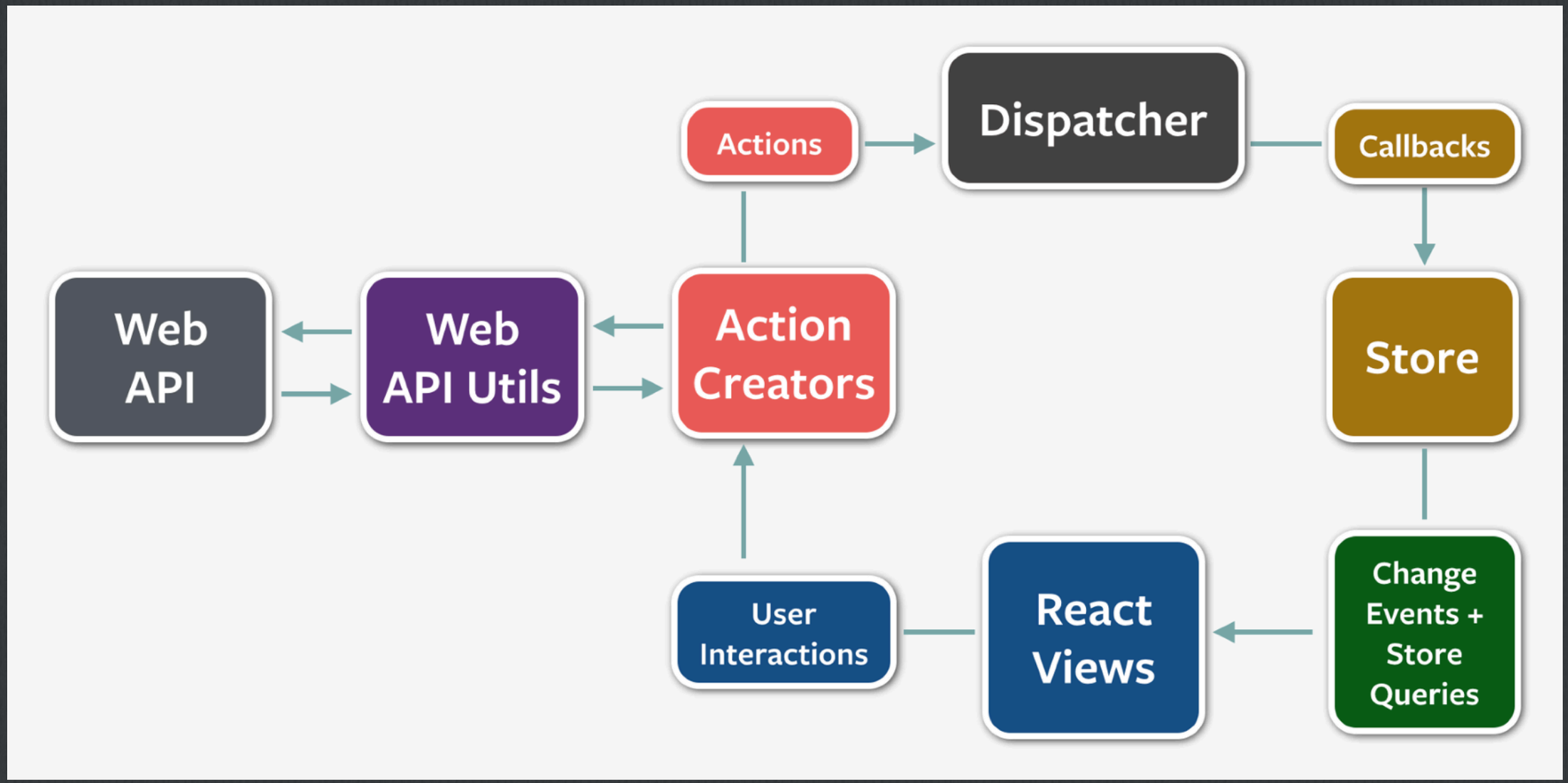
REACT-DEATHMATCH

- src
  - app
    - hero
      - api
        - actions.ts
        - epics.ts
        - reducer.ts
        - service.ts
      - heroDetail
      - heroDetailContainer
      - heroList
      - heroListContainer
      - heroListItem
    - index.ts
    - model.ts
    - routing.ts
  - store
    - epics.ts
    - model.ts
    - reducers.ts

ANGULAR-DEATHMATCH

- src
  - app
    - hero
      - api
        - actions.ts
        - epics.ts
        - reducer.ts
        - service.ts
      - heroDetail
      - heroDetailContainer
      - heroList
      - heroListContainer
      - heroListItem
    - model.ts
    - module.ts
    - routing.ts
  - store
    - epics.ts
    - model.ts
    - module.ts
    - reducers.ts

Redux Overview



Actions

```
export const LOAD_HEROES = 'LOAD_HEROES';
export const LOAD_STARTED = 'LOAD_STARTED';
export const LOAD_SUCCEEDED = 'LOAD_SUCCEEDED';
export const LOAD_FAILED = 'LOAD_FAILED';

/*
 * action creators
 */
export function loadHeroes(): HeroAPIAction {
  return {
    type: LOAD_HEROES,
    payload: [],
    meta: {}
  };
}
```

```
@Injectable()
export class HeroAPIActions {
  static readonly LOAD_HEROES = 'LOAD_HEROES';
  static readonly LOAD_STARTED = 'LOAD_STARTED';
  static readonly LOAD_SUCCEEDED = 'LOAD_SUCCEEDED';
  static readonly LOAD_FAILED = 'LOAD_FAILED';

  @dispatch()
  loadHeroes = (): HeroAPIAction => ({
    type: HeroAPIActions.LOAD_HEROES,
    payload: [],
    meta: {}
  });
```

Reducers

```
import { HeroAPIAction, LOAD_STARTED, LOAD_SUCCEEDED, LOAD_FAILE
import { IHeroList } from '../model';
import { Action } from 'redux';

const INITIAL_STATE: IHeroList = {
  heroes: [],
  loading: false,
  error: null,
};

export function createHeroAPIReducer() {
  return function heroReducer(state: IHeroList = INITIAL_STATE,
    a: Action): IHeroList {

    const action = a as HeroAPIAction;

    switch (action.type) {
      case LOAD_STARTED:
        return {
          ...state,
          heroes: [],
          loading: true,
          error: null,
        };
      case LOAD_SUCCEEDED:
        return {
          ...state,
          heroes: action.payload,
          loading: false,
          error: null,
        };
      case LOAD_FAILED:
        return {
          ...state,
          heroes: [],
          loading: false,
          error: action.error,
        };
    }

    return state;
  };
}
```

```
import { HeroAPIAction, HeroAPIActions } from './actions';
import { IHeroList } from '../model';
import { Action } from 'redux';

const INITIAL_STATE: IHeroList = {
  heroes: [],
  loading: false,
  error: null,
};

export function createHeroAPIReducer() {
  return function heroReducer(state: IHeroList = INITIAL_STATE,
    a: Action): IHeroList {

    const action = a as HeroAPIAction;

    switch (action.type) {
      case HeroAPIActions.LOAD_STARTED:
        return {
          ...state,
          heroes: [],
          loading: true,
          error: null,
        };
      case HeroAPIActions.LOAD_SUCCEEDED:
        return {
          ...state,
          heroes: action.payload,
          loading: false,
          error: null,
        };
      case HeroAPIActions.LOAD_FAILED:
        return {
          ...state,
          heroes: [],
          loading: false,
          error: action.error,
        };
    }

    return state;
  };
}
```

Epics

```
15  const heroesNotAlreadyFetched = (
16    state: IAppState): boolean => !(
17      state &&
18      state.heroes &&
19      state.heroes.heroes.length);
20
```

```
21  export default function createLoadHeroEpic(): Epic<HeroAPIAction, IAppSta
22      var service = new HeroAPIService();
23      return (action$, store) => action$
24        .ofType(LOAD_HEROES)
25        .filter(() => heroesNotAlreadyFetched(store.getState()))
26        .switchMap(() => service.getAll()
27          .map(data => loadSucceeded(data))
28          .catch(response => of(loadFailed({
29            status: '' + response.status,
30          })))
31          .startWith(loadStarted()));
32    }
```

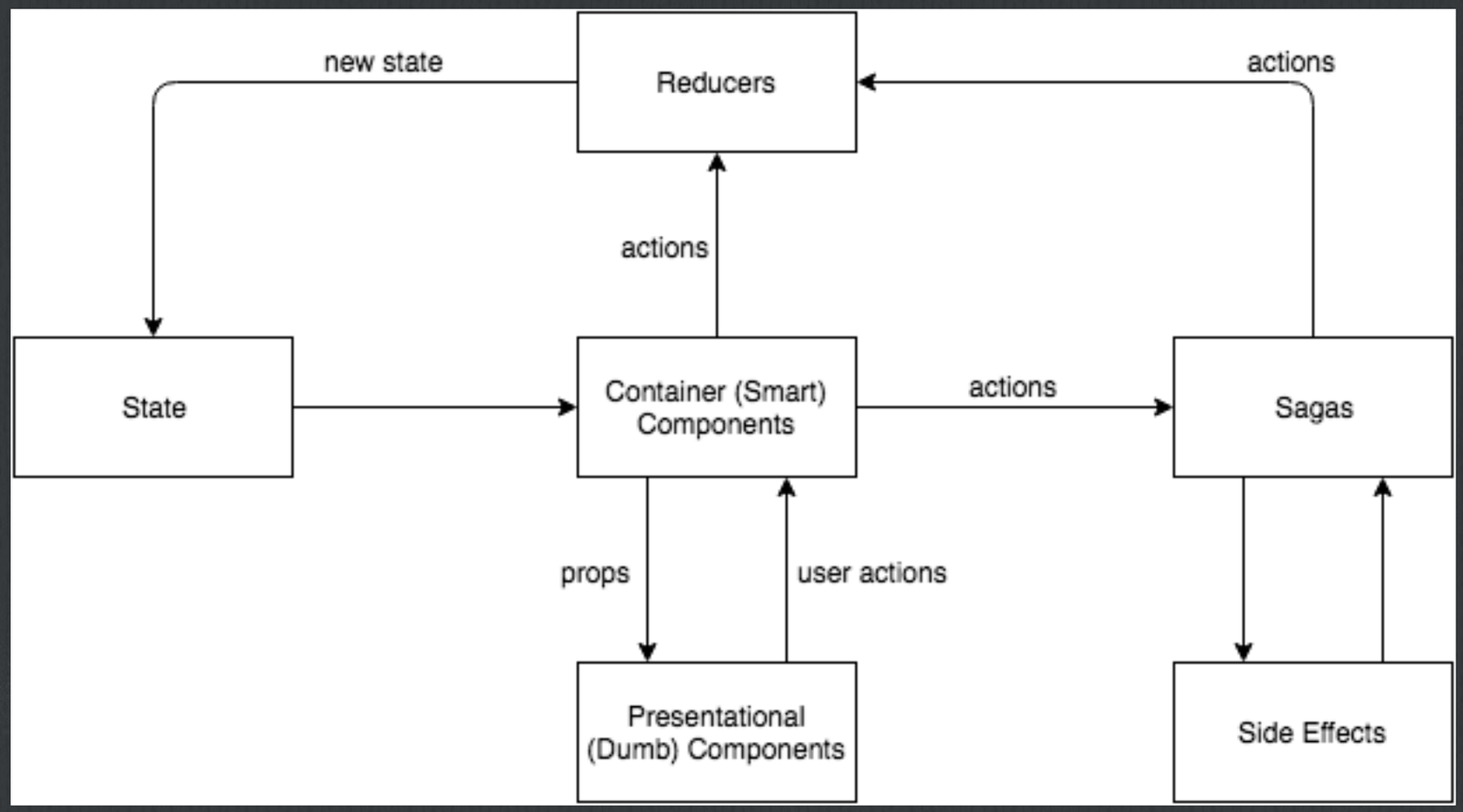
```
13  const heroesNotAlreadyFetched = (
14    state: IAppState): boolean => !(
15      state &&
16      state.heroes &&
17      state.heroes.heroes.length);
18
19  @Injectable()
20  export class HeroAPIEpics {
21    constructor(
22      private service: HeroAPIService,
23      private actions: HeroAPIActions,
24    ) {}
25
26    public createEpic() {
27      return createEpicMiddleware(this.createLoadHeroEpic());
28    }
29
30    private createLoadHeroEpic(): Epic<HeroAPIAction, IAppState> {
31      return (action$, store) => action$
32        .ofType(HeroAPIActions.LOAD_HEROES)
33        .filter(() => heroesNotAlreadyFetched(store.getState()))
34        .switchMap(() => this.service.getAll()
35          .map(data => this.actions.loadSucceeded(data))
36          .catch(response => of(this.actions.loadFailed({
37            status: '' + response.status,
38          })))
39          .startWith(this.actions.loadStarted()));
40    }
41  }
```

API Service

```
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/observable/fromPromise';
import { IHero, fromServer } from '../model';

// The heroes API we created.
const URL = 'https://angular-1-training-class-api.herokuapp.com/heroes';

export class HeroAPIService {
  constructor() {}

  getAll = (): Observable<IHero[]> => {
    return Observable.fromPromise(fetch(URL, {
        method: 'GET'
    }).then(resp => resp.json()))
    .map(records => records.map(fromServer));
  }

}
```

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/map';
import 'rxjs/add/observable/of';

import { IHero, fromServer } from '../model';

// The heroes API we created.
const URL = 'https://angular-1-training-class-api.herokuapp.com/heroes';

@Injectable()
export class HeroAPIService {
  constructor(private http: Http) {}

  getAll = (): Observable<IHero[]> =>
    this.http.get(URL)
      .map(resp => resp.json())
      .map(records => records.map(fromServer));
}
```

Smart/Dumb Components



```tsx
class HeroListContainerComponent extends React.Component<HeroListContainerComponentProps, {}> {

  changeHero(hero: IHero) {
    this.props.history.push('/hero/' + hero.id);
  }

  componentDidMount() {
    this.props.loadHeroes();
  }

  render() {
    const {
      heroes
    } = this.props;
    const sortedHeroes = sortHeroes(heroes);
    return (
      <HeroListComponent heroes={sortedHeroes} onSelect={(hero) => this.changeHero(hero)}></HeroListComponent>
    );
  }
}
```

HeroListContainer

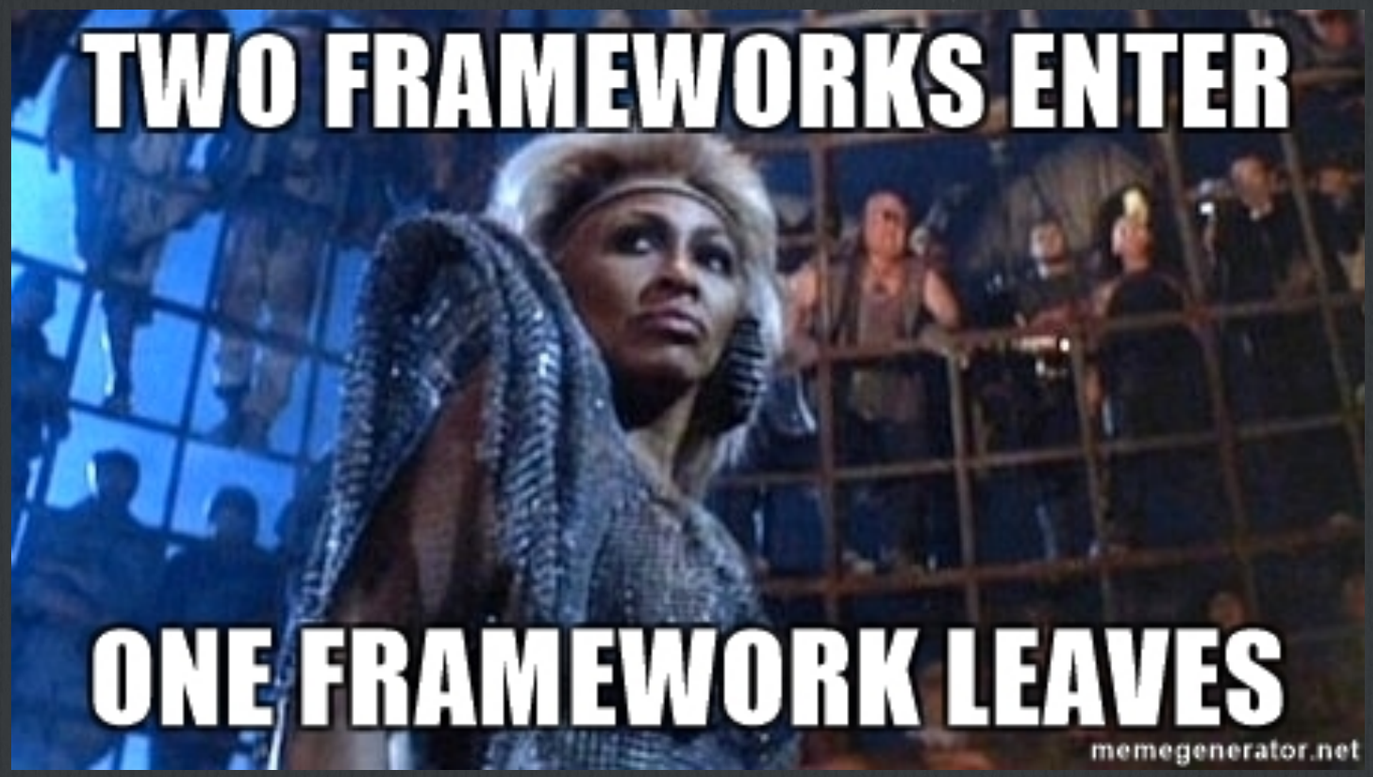
```ts
@Component({
  selector: 'hero-list-container',
  template: '<hero-list [heroes]="heroes$ | async" (onSelect)="changeHero($event, hero)"></hero-list>'
})
export class HeroListContainerComponent {

  // Get data out of the Redux store as observables.
  @select$(['heroes', 'heroes'], sortHeroes)
  readonly heroes$: Observable<IHero[]>;

  changeHero(hero: IHero) {
    this.router.navigate(['/hero', hero.id]);
  }

  constructor(
    actions: HeroAPIActions,
    private router: Router
  ) {
    actions.loadHeroes();
  }
}
```

```tsx
export function HeroListComponent({
    heroes,
    onSelect = (hero: IHero) => null
} : HeroListComponentProps) {

  const item_list = heroes.map((hero) => {
    return (
      <HeroListItemComponent key={hero.id} hero={hero} onSelect={(hero) => onSelect(hero)}></HeroListItemComponent>
    );
  });

  return (
    <div className="row">
        <div className="col col-12">
            {item_list}
        </div>
    </div>
  );

}
```

HeroList

```ts
@Component({
  selector: 'hero-list',
  templateUrl: './index.html',
  styleUrls: ['./index.css'],
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class HeroListComponent {

  @Input() heroes: IHero[];
  @Output() onSelect = new EventEmitter<IHero>();

  onHeroSelected(hero: IHero) {
    this.onSelect.emit(hero);
  }

}
```

```html
<div *ngFor="let hero of heroes">
  <hero-list-item [hero]="hero" (onSelect)="onHeroSelected(hero)"></hero-list-item>
</div>
```


TWO FRAMEWORKS ENTER
ONE FRAMEWORK LEAVES
memegenerator.net

How do we break this down?

- ☐ How are they similar?
- ☐ What are the deciding factors?
- ☐ Who would choose each?

Similar

Typescript

Redux

CLI

Flux Standard Actions

Routing

Observables

Epics

Yarn

Webpack 2

Redux Dev Tools

Deciding factors

- ☐ Closer to vanilla - React
- ☐ What is in core project vs. what is not - big = Angular, small = React
- ☐ Does anybody want Typescript? - built in Angular, mostly supported React
- ☐ Forms - Angular
- ☐ What about Native? - React Native is great and more mature
- ☐ Documentation/pace of change - Angular docs way better, React moving fast

Who should choose Angular?

- ☐ Big corporate teams - Typescript, all in one tools
- ☐ Beginners - documentation is solid, less libs
- ☐ Form-heavy projects
- ☐ Simple mobile apps

Who should choose React?

- [ ] Startups who don't want to be constrained
  - [ ] React Native
  - [ ] Closer to pure JS
- [ ] Junior devs making the jump
- [ ] Everyone else

Thanks! Connect with Me!
We would love to build your next app

A Software Presentation From ZERRTECH

Jeremy Zerr

Site: http://www.zerrtech.com
LinkedIn: http://www.linkedin.com/in/jrzerr
Twitter: http://www.twitter.com/jrzerr