INTRODUCTION TO

REACT

ABOUT AMBROSE

- Glad dad of 6 munchkins.
- Making pro software for ~18 years
  Computer geek since 1987
- Shipped over 45 projects, led over 20. 2 Patents.
  Sometime dev, arch, UX guy, manager, PM, etc.
- 8x Microsoft MVP; Books, Blogs…
- Gone Skinning Dipping in Bulgarian Mountain Hot Springs

THE PLAN

- Why MobX?
  In Which I Cover the Complete History of Functional State Management (j/k)
- Key Concepts
  A Bit Mind-bending but Easy to Use
- With the Showing of Some Examples
  Someone just told me these are the BEST examples!

ABOUT YOU?

- Front End or Full Stack Dev
- Comfy with JavaScript, ideally ES2015+.
- You went to my last session or already are familiar with React.
- You are not a MobX expert.

WHY MOBX?

- It Just Feels Right
- Easy to Use*
- Less Trouble than setState
- Simpler/Less Dogmatic then Redux
- Widely Used https://github.com/mobxjs/awesome-mobx

PREFACE: GETTING MOBX

- https://reactjs.org/ <- First Get React
- https://mobx.js.org/ - Then Get MobX

```
npm i mobx mobx-react —save
```

KEY CONCEPT #1

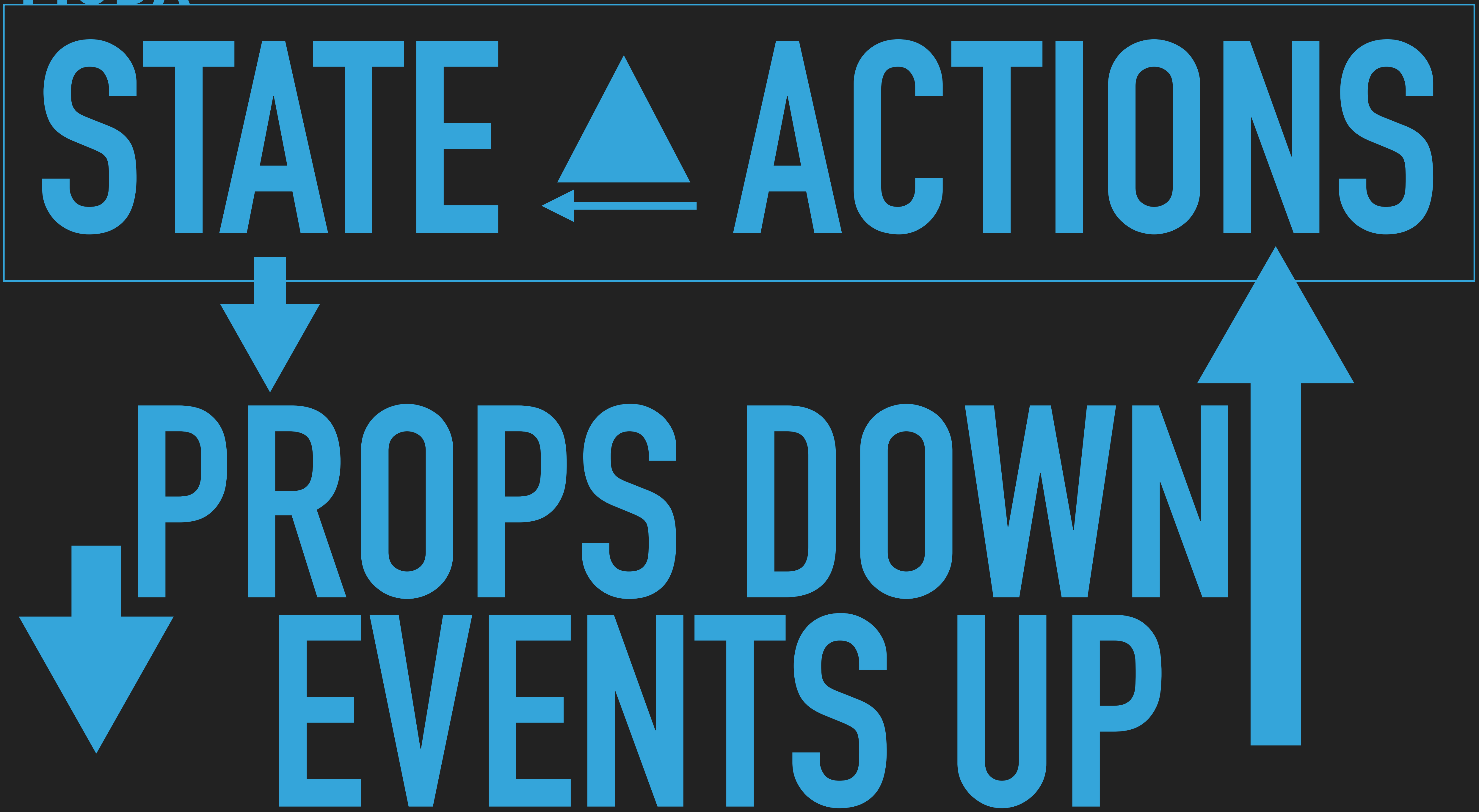
“UNIDIRECTIONAL DATA FLOW”

COMPONENT COMMUNICATION

↓ PROPS DOWN

EVENTS UP ↑

MOBX

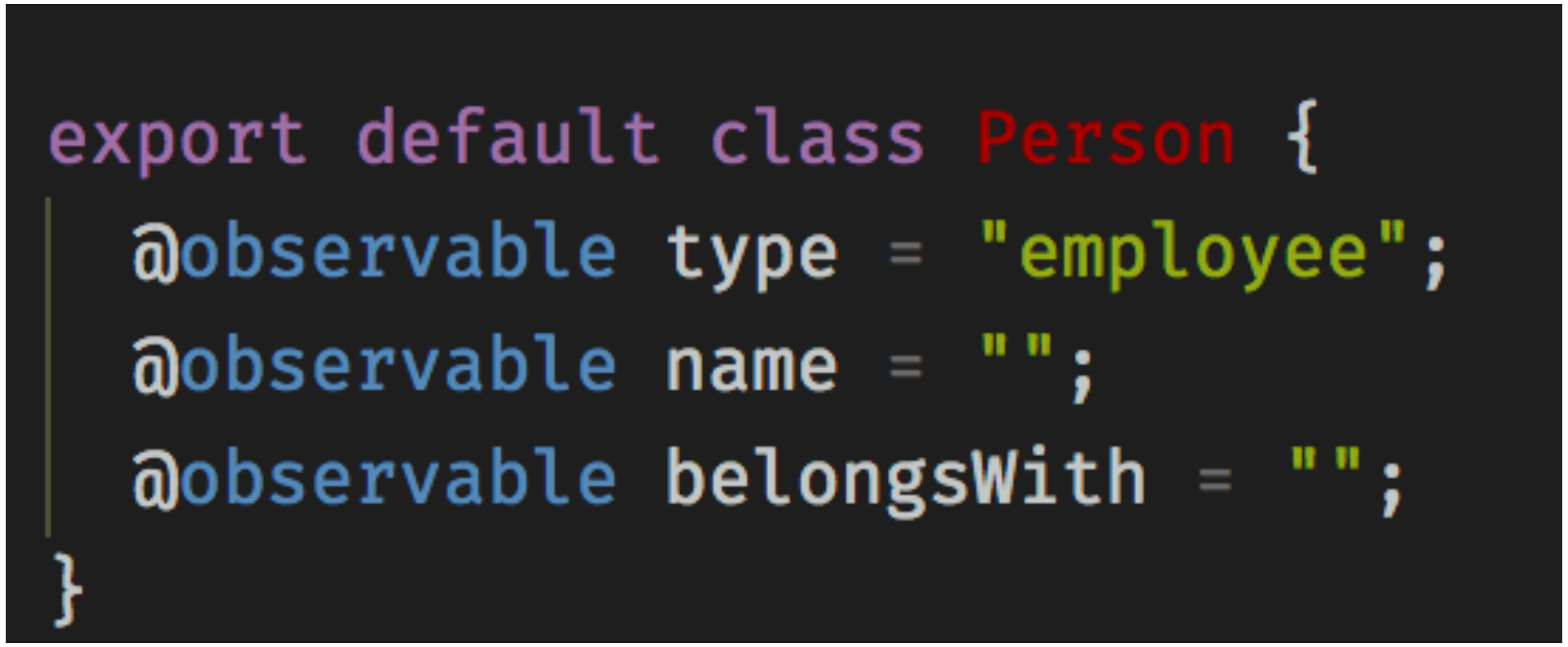


KEY CONCEPT #2

OBSERV<u>ABLE</u>

JUST ADD WATER!

- Add MobX to your project.
- Start decorating!



```
export default class Person {
  @observable type = "employee";
  @observable name = "";
  @observable belongsWith = "";
}
```

KEY CONCEPT #2.5

DECORATORS

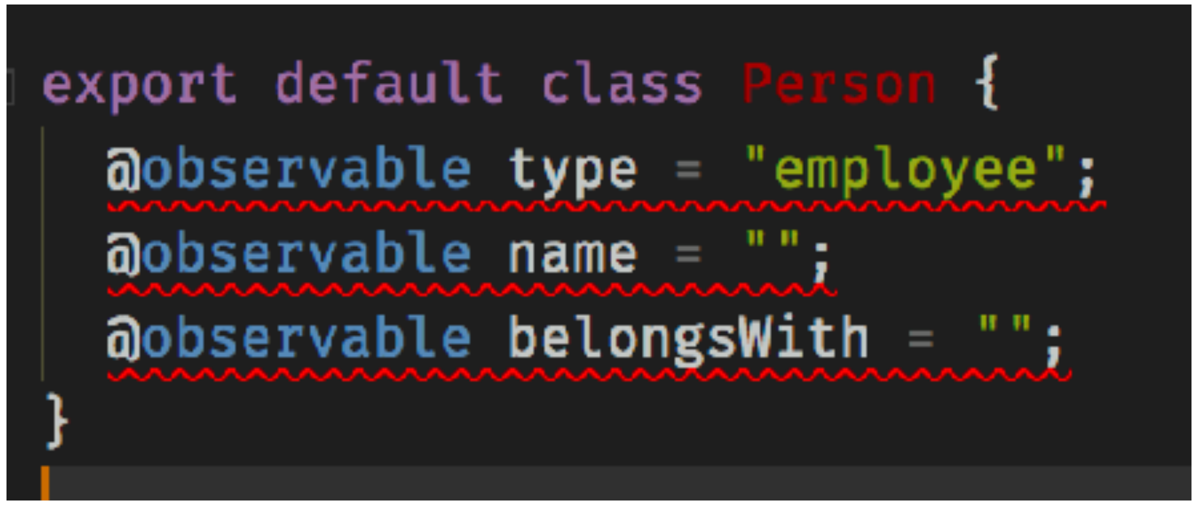
DECORATE YOUR CLASSES (FUNCTIONS) AND PROPERTIES

- Declarative way to augment your code. (Similar to .NET Attributes)
- Syntax*: prefix thing:
  `@[decorator][(options)] [thing]`
- Very common for MobX:
  - `@observable myProp;`
  - `@observer class MyComponent {}`

*Syntax is subject to change as this is a Stage 2 proposal.

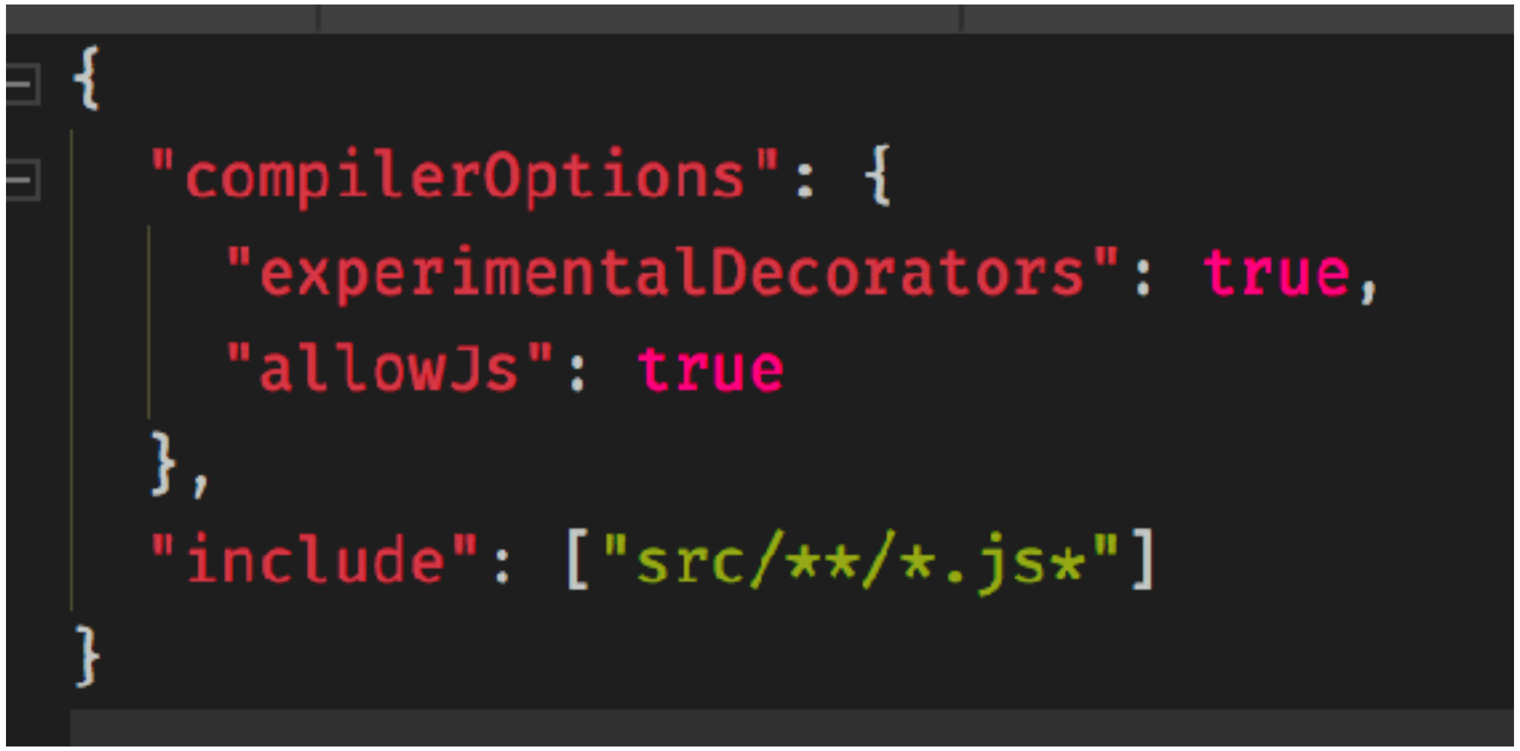
NB: USING DECORATORS WITH VS CODE

- Unless you like this:



```
export default class Person {
  @observable type = "employee";
  @observable name = "";
  @observable belongsWith = "";
}
```

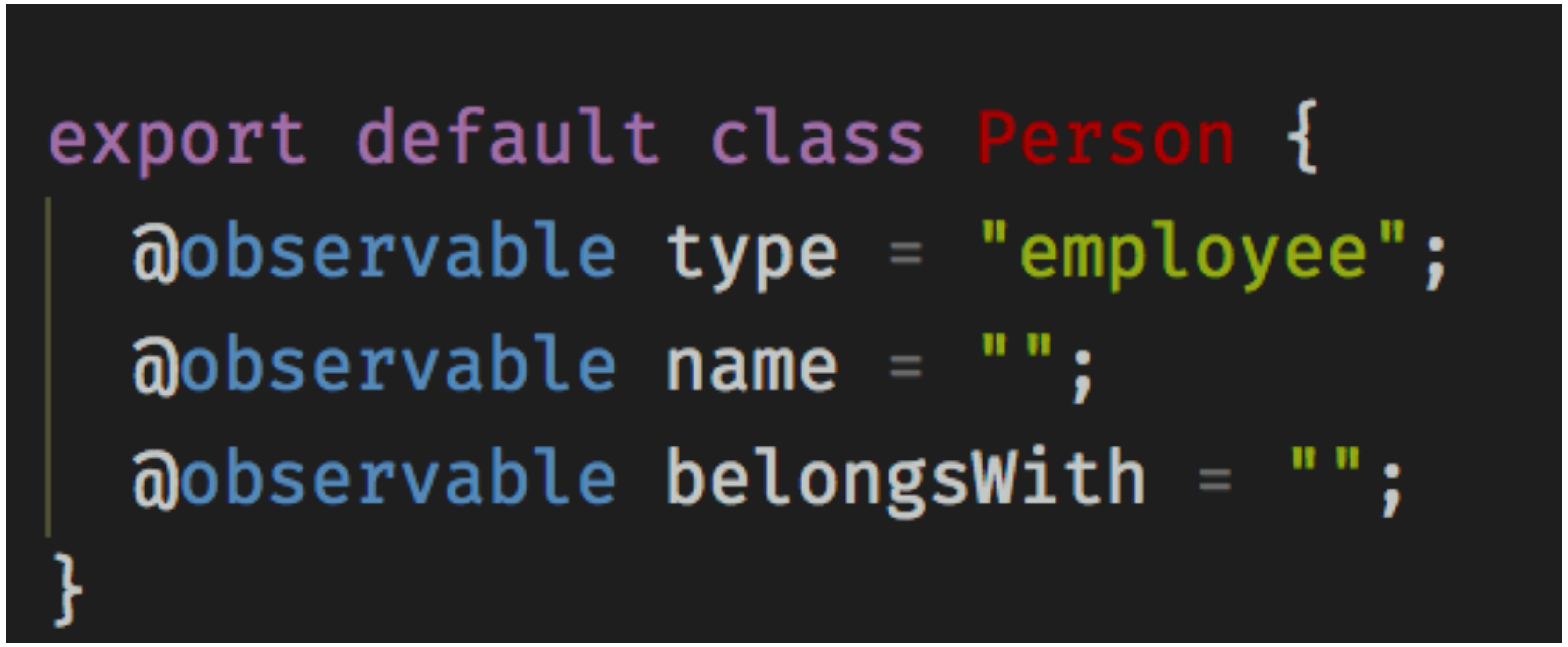
- Add a **`tsconfig.json`** to your project root like this:



```
{
  "compilerOptions": {
    "experimentalDecorators": true,
    "allowJs": true
  },
  "include": ["src/**/*.js*"]
}
```

BACK TO JUST ADDING WATER!

- Add MobX to your project.
- Start decorating!



```
export default class Person {
  @observable type = "employee";
  @observable name = "";
  @observable belongsWith = "";
}
```

DON'T FORGET ABOUT @COMPUTED!

Ambrose

CAUTION: MOBX AND CREATE-REACT-APP

- **CRA - Does not allow build config. Does not allow decorators (Stage 2).**
- Option: Use without decorators https://github.com/benawad/simple-mobx/tree/1_counter
- Option: Use `npm run eject` https://swizec.com/blog/mobx-with-create-react-app/swizec/7158
- Option: Use custom-react-scripts https://medium.com/@kitze/configure-create-react-app-without-ejecting-d8450e96196a
- More on with/without decorators: https://mobx.js.org/best/decorators.html

CONVERTING CREATE-REACT-APP

- `yarn remove react-scripts && yarn add custom-react-scripts`
- Add .env file with `REACT_APP_DECORATORS = true;`
- `npm i mobx mobx-react –save`

KEY CONCEPT #3

OBSERVER

KEY CONCEPT #4

MODIFY THROUGH “ACTIONS”

KEY CONCEPT #5

THE "STORE"

https://mobx.js.org/best/store.html

KEY CONCEPT #6

OTHER “DERIVATIONS” AND GOODIES

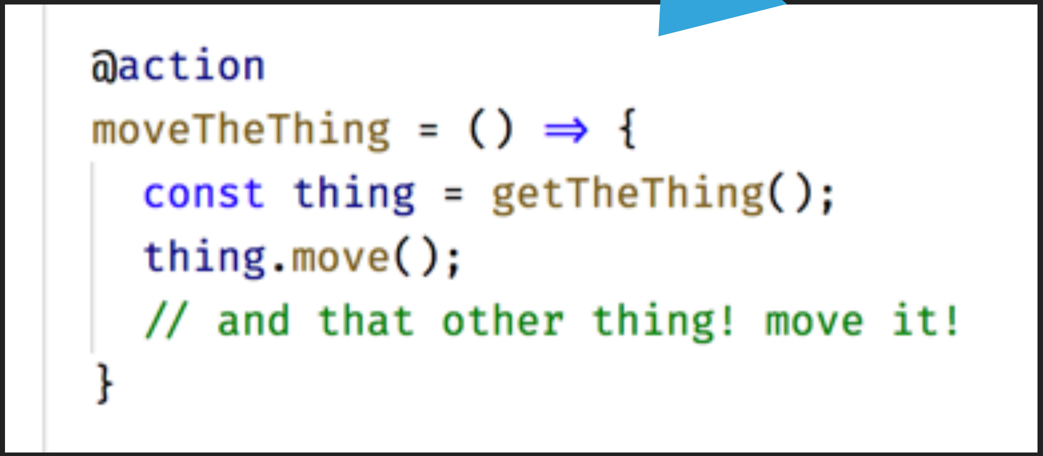
More at: https://mobx.js.org/intro/concepts.html

COMPUTED

HEY, I PROBABLY HAVE A NEW VALUE!

ONE OF MY OBSERVABLES CHANGED

REACTION



```
@action
moveTheThing = () => {
  const thing = getTheThing();
  thing.move();
  // and that other thing! move it!
}
```

ONE OF MY OBSERVABLES CHANGED

OTHER GOODIES

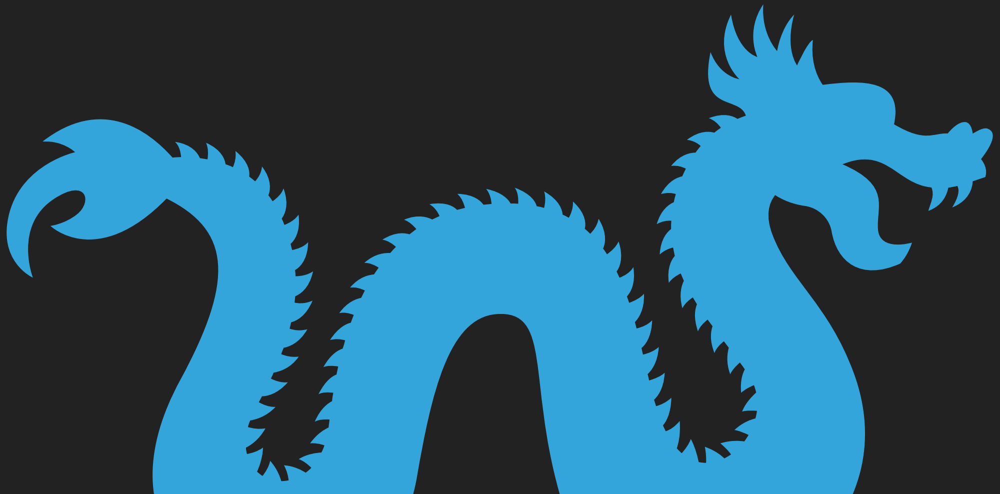
https://mobx.js.org/refguide/api.html

Scroll down to Utilities

isArrayLike!

WARNING:

THERE BE DRAGONS



THAT'S IT! YOU'VE SEEN…

- What MobX is All About
- Unidirectional Data Flow
- Decorators:
  - @observable
  - @observer
  - @computed
- How to Get More Reactive
- Modifying State with Actions
- Fancypants "Stores"
- Other Goodies

REPETITIO MATER STUDIORUM EST.

Ancient Dude

REPETITION IS THE MOTHER OF ALL LEARNING.

Dude What Speaks Modern English