JS Fundamentals for

Chris Aquino!

@radishmouse

- Director of Web Engineering @bignerdranch
- Instructor of Front-End Web Development bootcamp

Big Nerd Ranch

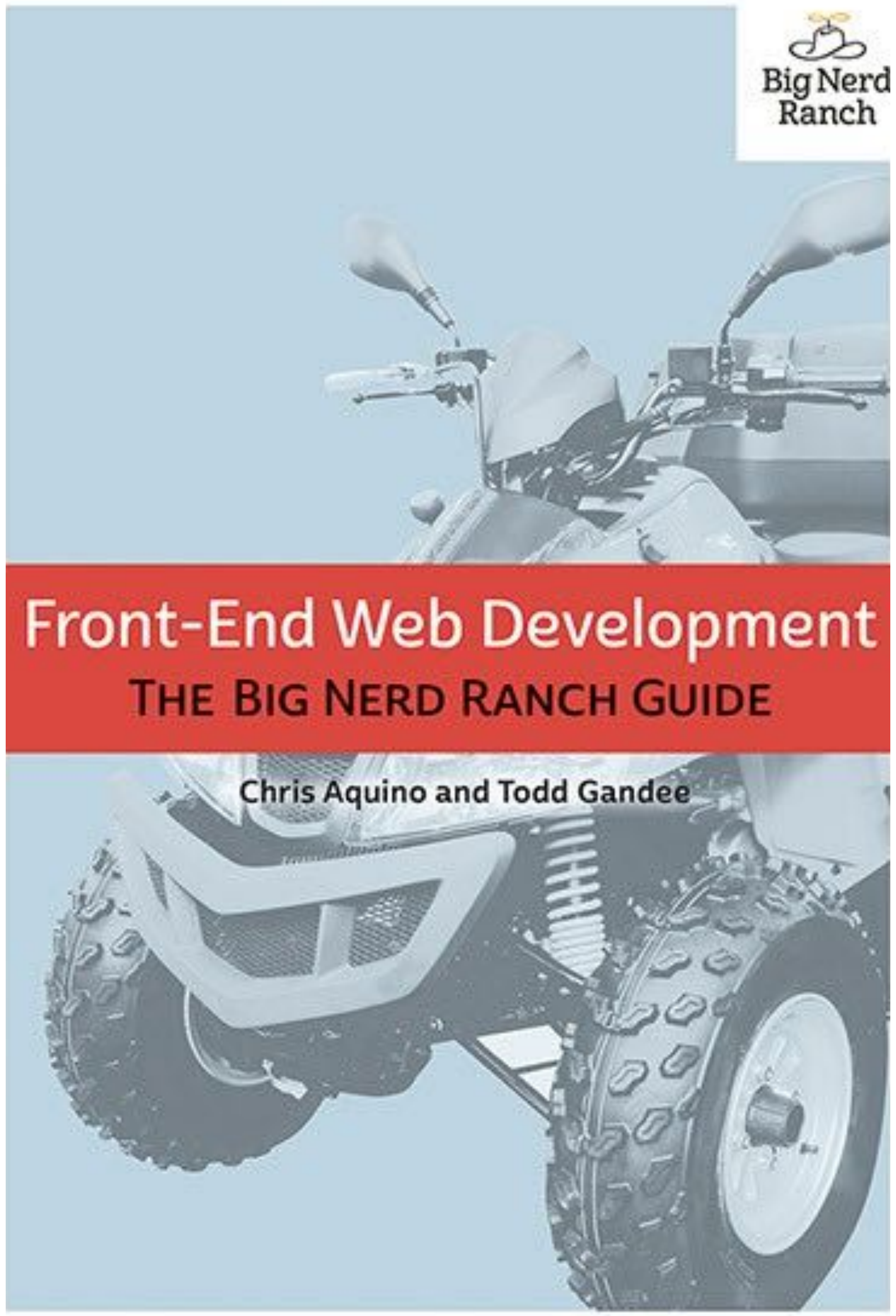
We develop.

We teach.

We write.

Aw yiss.

- come take my class
- come take my class
- come take my class
- or buy the book
- then take my class



THAT REACT
SO HOT RIGHT NOW
imgflip.com

Today, let’s demystify:

- Functional, declarative views
- Virtual DOM
- One-way data flow
- Component architecture
- Immutable data structures

“Learn once, write anywhere”

–Marketing person at Facebook

WebVR

JS

iOS

Five buckets o’ React

Yes, I made up this word.

Functions

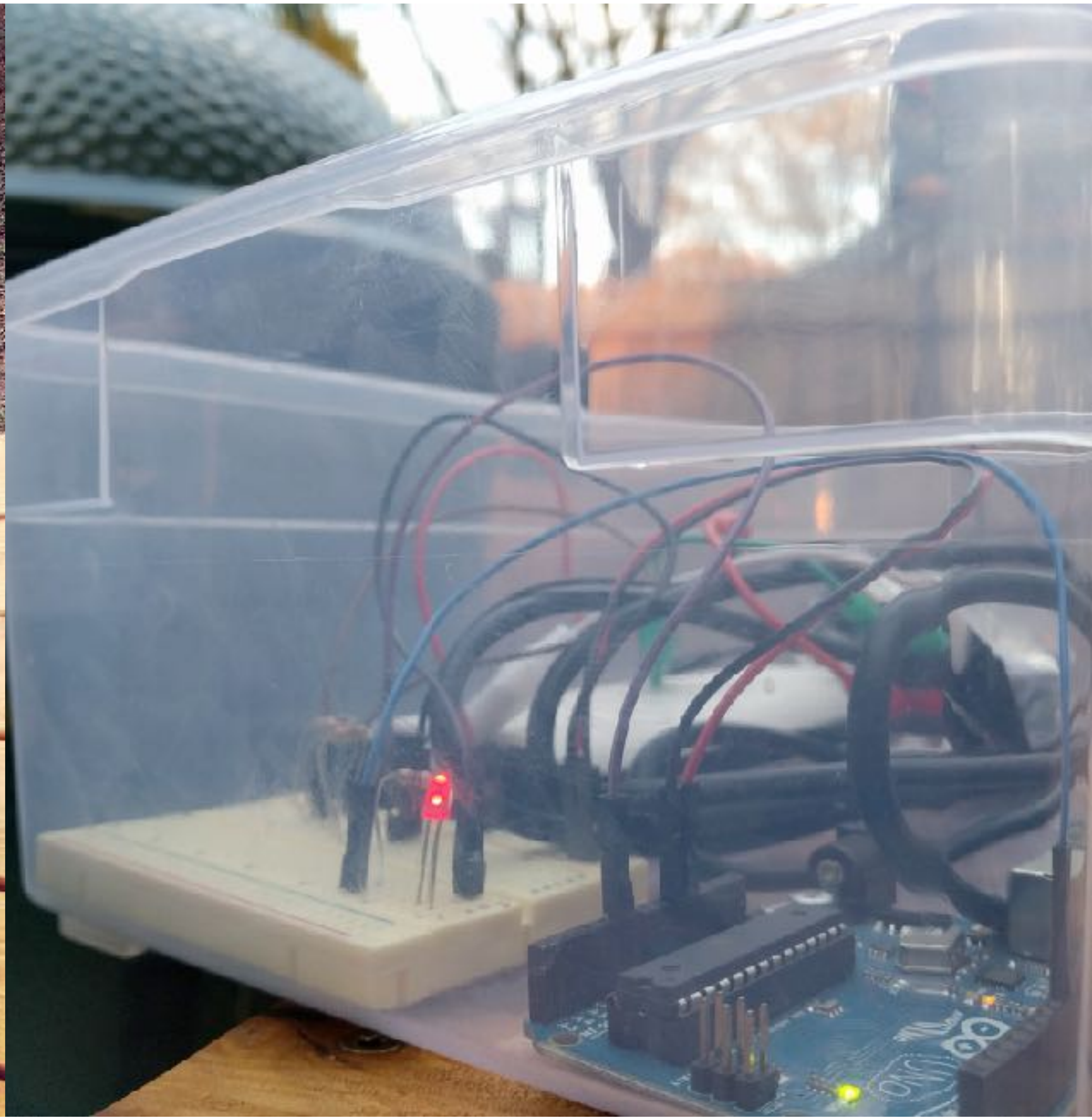
Objects

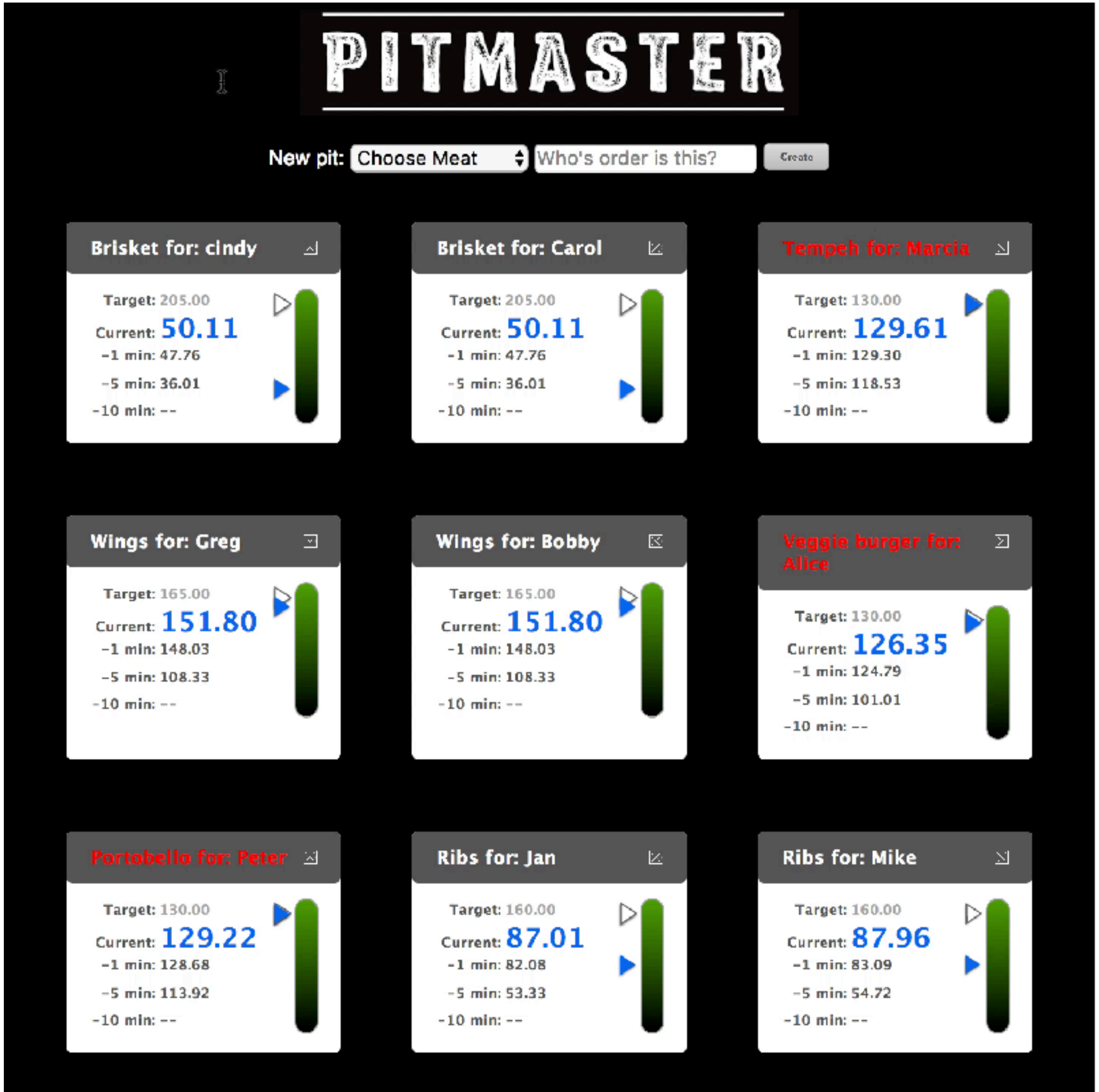
Classes

Modules

Immutables

PitMaster





PITMASTER

New pit: Choose Meat | Who's order is this? | Create

Brisket for: cindy
Target: 205.00
Current: 50.11
-1 min: 47.76
-5 min: 36.01
-10 min: --

Brisket for: Carol
Target: 205.00
Current: 50.11
-1 min: 47.76
-5 min: 36.01
-10 min: --

Tempeh for: Marcia
Target: 130.00
Current: 129.61
-1 min: 129.30
-5 min: 118.53
-10 min: --

Wings for: Greg
Target: 165.00
Current: 151.80
-1 min: 148.03
-5 min: 108.33
-10 min: --

Wings for: Bobby
Target: 165.00
Current: 151.80
-1 min: 148.03
-5 min: 108.33
-10 min: --

Veggie burger for: Alice
Target: 130.00
Current: 126.35
-1 min: 124.79
-5 min: 101.01
-10 min: --

Portobello for: Peter
Target: 130.00
Current: 129.22
-1 min: 128.68
-5 min: 113.92
-10 min: --

Ribs for: Jan
Target: 160.00
Current: 87.01
-1 min: 82.08
-5 min: 53.33
-10 min: --

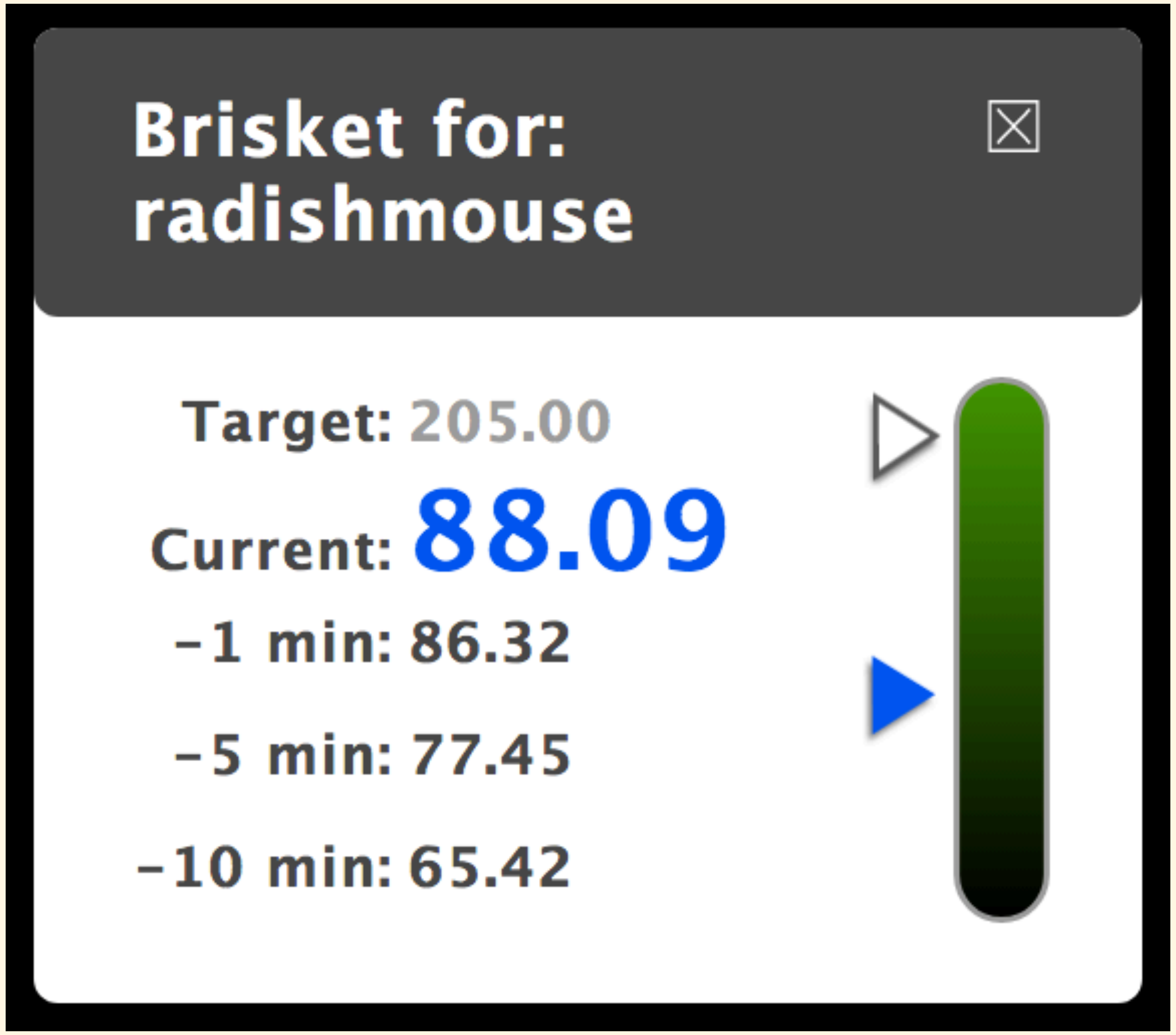
Ribs for: Mike
Target: 160.00
Current: 87.96
-1 min: 83.09
-5 min: 54.72
-10 min: --

```
import React from 'react';

import FoodChooserForm from './FoodChooserForm';
import Monitor from '../containers/Monitor';
import MonitorPanel from '../containers/MonitorPanel';

import {
  FOOD_CHOICES,
  tempsForFood,
  cookFactorForFood,
  ROOM_TEMP
} from '../config';

import {
  cookFood,
  Sensor
} from '../lib/GrillSimulator';

class PitMaster extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      orders: []
    };
  }

  componentWillUnmount() {
    this.state.orders.forEach(({sensor}) => sensor.stop());
  }

  render() {
    return (
      <div>
        <h1>PitMaster</h1>
        <FoodChooserForm
          foodChoices={FOOD_CHOICES}
          submitHandler={this._addOrder}
        />

        <MonitorPanel orderArray={
          this.state.orders.map((order) => ({
            key: order.id,
            name: order.orderName,
            food: order.foodChoice,
            foodTemperature: order.current,
            historyArray: order.history,
            ovenTemperature: tempsForFood(order.foodChoice).oven
          }))
        } />
      </div>
    );
  }

  _addOrder = (order) => {
    order.id = (new Date()).getTime();
    order.sensor = new Sensor(cookFood(ROOM_TEMP,
                                       tempsForFood(order.foodChoice).oven,
                                       cookFactorForFood(order.foodChoice)
                                      ),
                              () => this._updateTemperatures(order.id));
    this.setState({
      orders: [...this.state.orders, order]
    });
    order.sensor.start();
  }

  _updateTemperatures = (id) => {
    this.setState({
      orders: this.state.orders.map((order) => (
        order.id === id ? {
                            ...order,
                            current: order.sensor.current(),
                            history: [
                              order.sensor.minutesAgo(1),
                              order.sensor.minutesAgo(5),
                              order.sensor.minutesAgo(10),
                            ]
                          }
                        : order

      ))
    })
  }
}

export default PitMaster;
```

Functions

$$f(d) = v$$

–Tyler McGinnis



Brisket for:
radishmouse

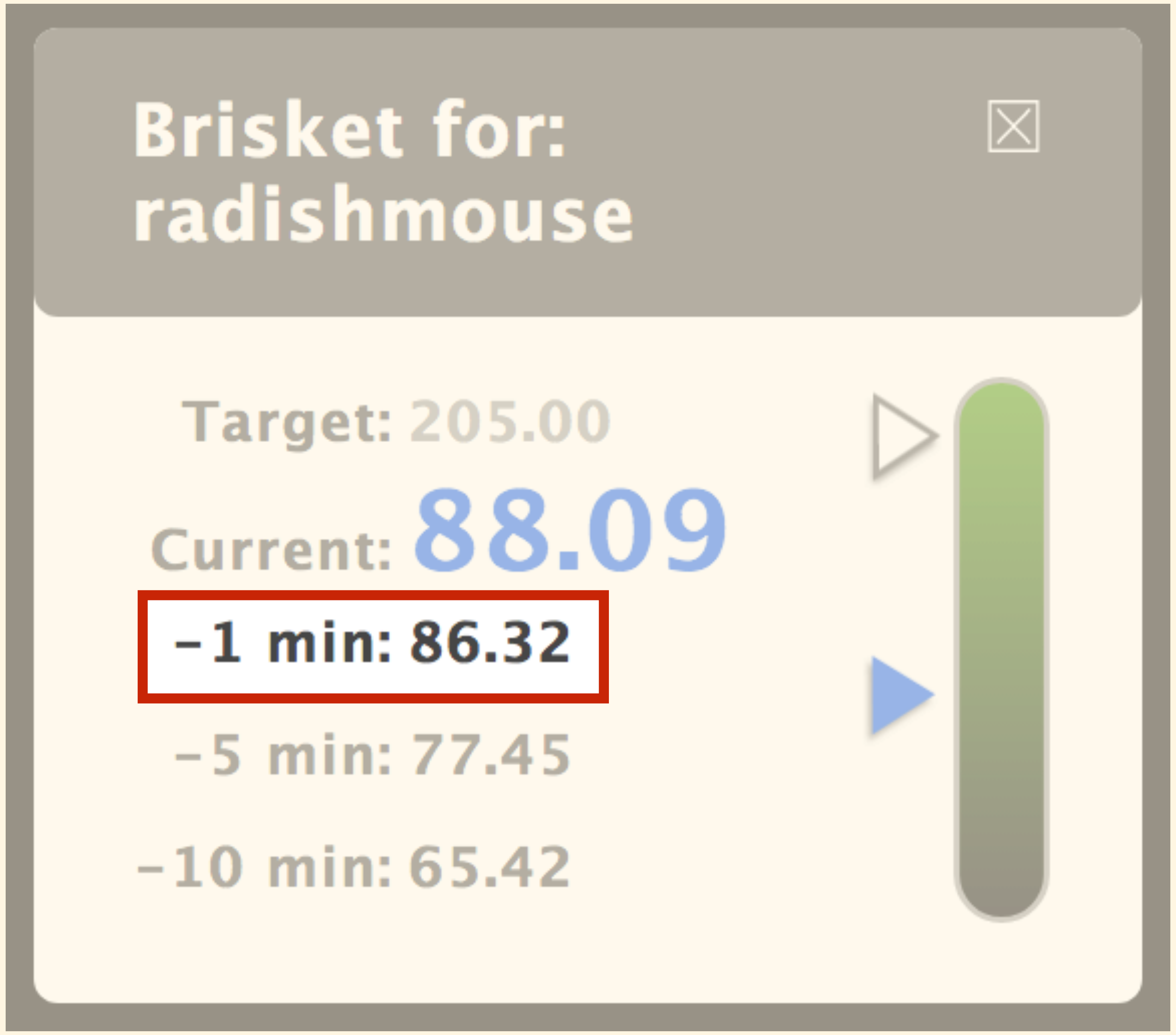
Target: 205.00

Current: 88.09

-1 min: 86.32

-5 min: 77.45

-10 min: 65.42



Brisket for:
radishmouse

Target: 205.00

Current: 88.09

-1 min: 86.32

-5 min: 77.45

-10 min: 65.42

```
function Readout(value) {
  return value.toFixed(2);
}
```

```
function Readout(value) {
  return value.toFixed(2);
}

Readout(98.675);
// 98.67
```

```
var Readout = (value) => {
  return value.toFixed(2);
};

Readout(98.675);
// 98.67
```

```
let Readout = (value) => {
  return value.toFixed(2);
};

Readout(98.675);
// 98.67
```

```
const Readout = (value) => {
  return value.toFixed(2);
};

Readout(98.675);
// 98.67
```

```
const Readout = (value) => {
  return value.toFixed(2);
};

const Readout = value => {
  return value.toFixed(2);
};

const Readout = (value) => value.toFixed(2);

const Readout = value => value.toFixed(2);

const Readout = (value) => (
  value.toFixed(2)
);
```

```
const Readout = (value) => (
  value.toFixed(2) + ' degrees F'
);

Readout(98.675);
// 98.67 degrees F
```

```
const Readout = (value) => (
  `${value.toFixed(2)} degrees F`
);

Readout(98.675);
// 98.67 degrees F
```

```
const Readout = (value) => {
  value = value || 0;
  return `${value.toFixed(2)} degrees F`;
};

Readout();
// 0.00 degrees F

Readout(98.675);
// 98.67 degrees F
```

```
const Readout = (value=0) => (
  `${value.toFixed(2)} degrees F`
);
```

```
const Readout = (value=0) => (
  degreesF(value)
);

const degreesF = (temperature) => (
  `${temperature.toFixed(2)} degrees F`
);


Readout(98.675);
// 98.67 degrees F

Readout();
// 0.00 degrees F
```

```
const Readout = (formatterFn, value=0) => (
  formatterFn(value)
)

const degreesF = (temperature) => (
  `${temperature.toFixed(2)} degrees F`
)

Readout(degreesF, 98.675);
// 98.67 degrees F

Readout(degreesF);
// 0.00 degrees F
```

```
const Readout = (formatterFn, value=0) => {
  if (typeof formatterFn === 'function') {
    return formatterFn(value);
  } else {
    return value;
  }
};

const degreesF = (temperature) => (
  `${temperature.toFixed(2)} degrees F`
);

Readout(degreesF, 98.675);
// 98.67 degrees F

Readout(degreesF);
// 0.00 degrees F

Readout(undefined, 98.675);
// 98.675

Readout();
// 0
```

```
const Readout = (formatterFn, value=0) => (
  typeof formatterFn === 'function' ? formatterFn(value)
                                    : value
);

const degreesF = (temperature) => (
  `${temperature.toFixed(2)} degrees F`
);
```

```
const Readout = (formatterFn, value=0) => (
  typeof formatterFn === 'function' ? formatterFn(value)
                                    : value
);

const degreesF = (temperature) => (
  `${temperature.toFixed(2)} degrees F`
);

const div = (content, className) => (
  `<div class="${className}">${content}</div>`
);

const span = (content, className) => (
  `<span class="${className}">${content}</span>`
);


const TemperaturePanel = (data) => (
  div(span(Readout(degreesF, data), 'green'), 'panel')
);


TemperaturePanel(92.675);

// <div class="panel"><span class="green">98.67 degrees F</span></div>
```

```
const TemperaturePanel = (data) => (
  div(span(Readout(degreesF, data), 'green'), 'panel')
);
```

```
const TemperaturePanel = ({data}) => (
  <div className='panel'>
    <span className='green'>
      <Readout
        formatterFn={degreesF}
        value={data}
      />
    </span>
  </div>
);
```

JSX!

```
const TemperatureHistory = ({temperatureArray}) => (
  <div>
    <h1>
      <TemperaturePanel data={temperatureArray[0]} />
    </h1>
    <TemperaturePanel data={temperatureArray[1]} />
    <TemperaturePanel data={temperatureArray[2]} />
  </div>
);
```

JSX

- XML description of nested function calls
- Transformed by the React library into function calls
- “Declarative” - resembles the resulting HTML
- Can be HTML elements or custom components

Functions

- JSX looks like XML, but really just functions
- In React, functions produce UI Components
- Get used to the ternary operator (? :)
- Default values are your friend
- const + arrow functions are a thing

```
<Readout data={98.675} />
```

Objects

```
const TemperaturePanel = ({data}) => (
  <div className='panel'>
    <span className='green'>
      <Readout
        formatterFn={degreesF}
        value={data}
      />
    </span>
  </div>
);
```

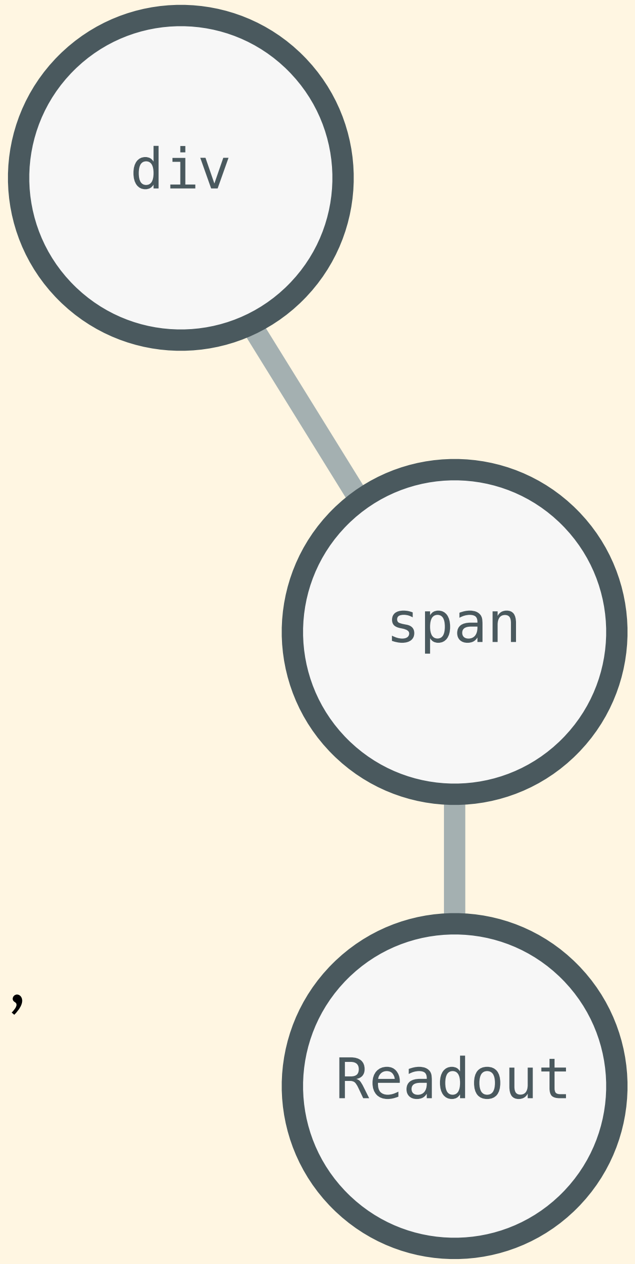
```
function TemperaturePanel(data) {
  return React.createElement(
    'div',
    { className: 'panel' },
    React.createElement(
      'span',
      { className: 'green' },
      React.createElement(Readout, {
        formatterFn: degreesF,
        value: data
      })
    )
  );
};
```

```
{
  type: "div",
  props: {
    className: "panel",
    children: [
      {
        type: "span",
        props: {
          className: "green",
          children: [
            {
              type: Readout,
              props: {
                formatterFn: degreesF,
                value: 98.675
              }
            }
          ]
        }
      }
    ]
  }
}
```

```
{
  type: "div",
  props: {
    className: "panel",
    children: [
      {
        type: "span",
        props: {
          className: "green",
          children: [
            {
              type: Readout,
              props: {
                formatterFn: degreesF,
                value: 98.675
              }
            }
          ]
        }
      }
    ]
  }
}
```

```
{
  type: "div",
  props: {
    className: "panel",
    children: [
      {
        type: "span",
        props: {
          className: "green",
          children: [
            {
              type: Readout,
              props: {
                formatterFn: degreesF,
                value: 98.675
              }
            }
          ]
        }
      }
    ]
  }
}
```

```
{
  type: "div",
  props: {
    className: "panel",
    children: [
      {
        type: "span",
        props: {
          className: "green",
          children: [
            {
              type: Readout,
              props: {
                formatterFn: degreesFn,
                value: 102.34
              }
            }
          ]
        }
      }
    ]
  }
}
```



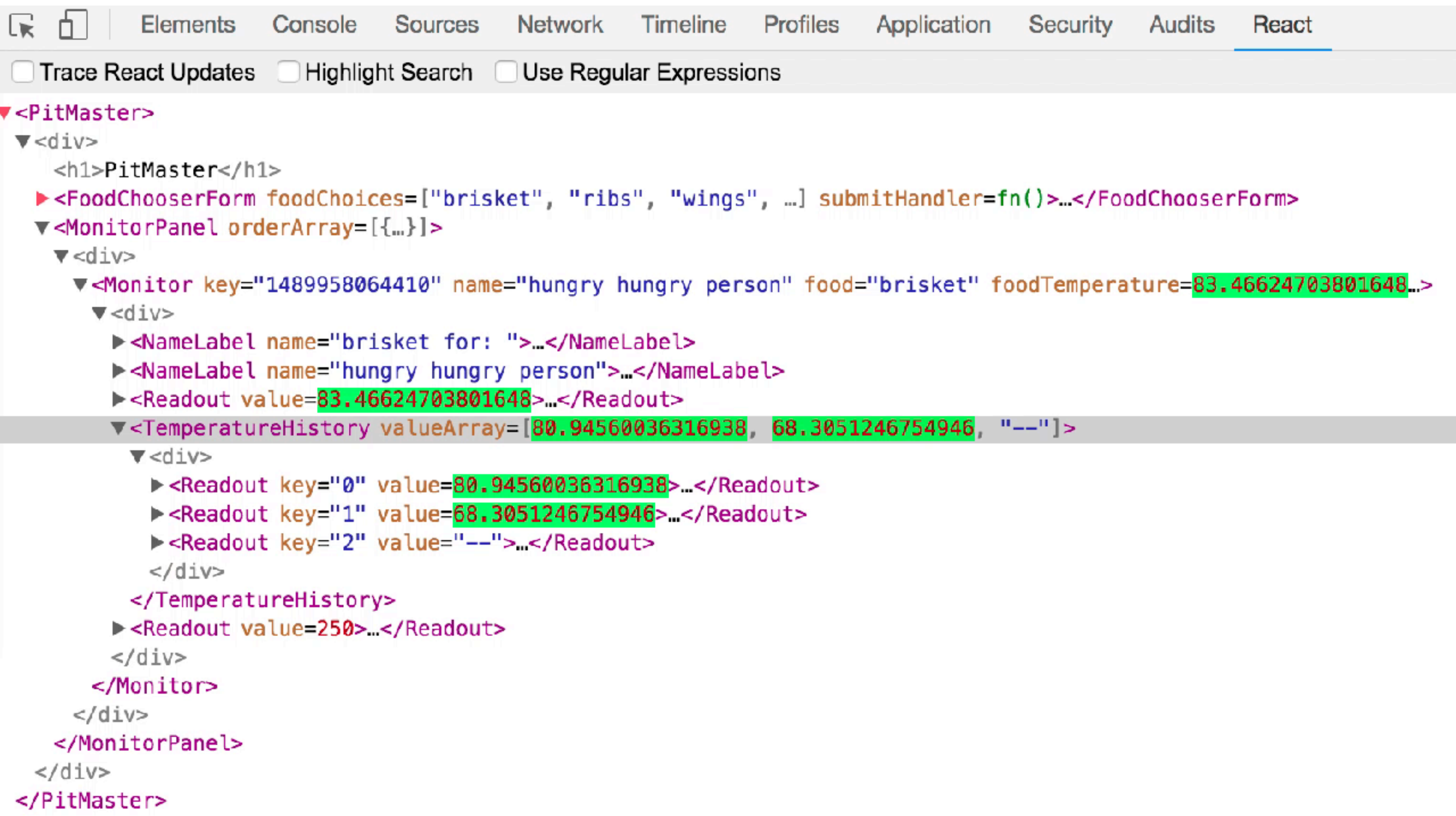
JSX renders to an Object

- JSX gets converted to React.createElement
- React.createElement returns a plain JavaScript object
- That Object is a description of the UI, including the data
- The UI only shows data that is passed down as a prop, starting at the top of the element tree

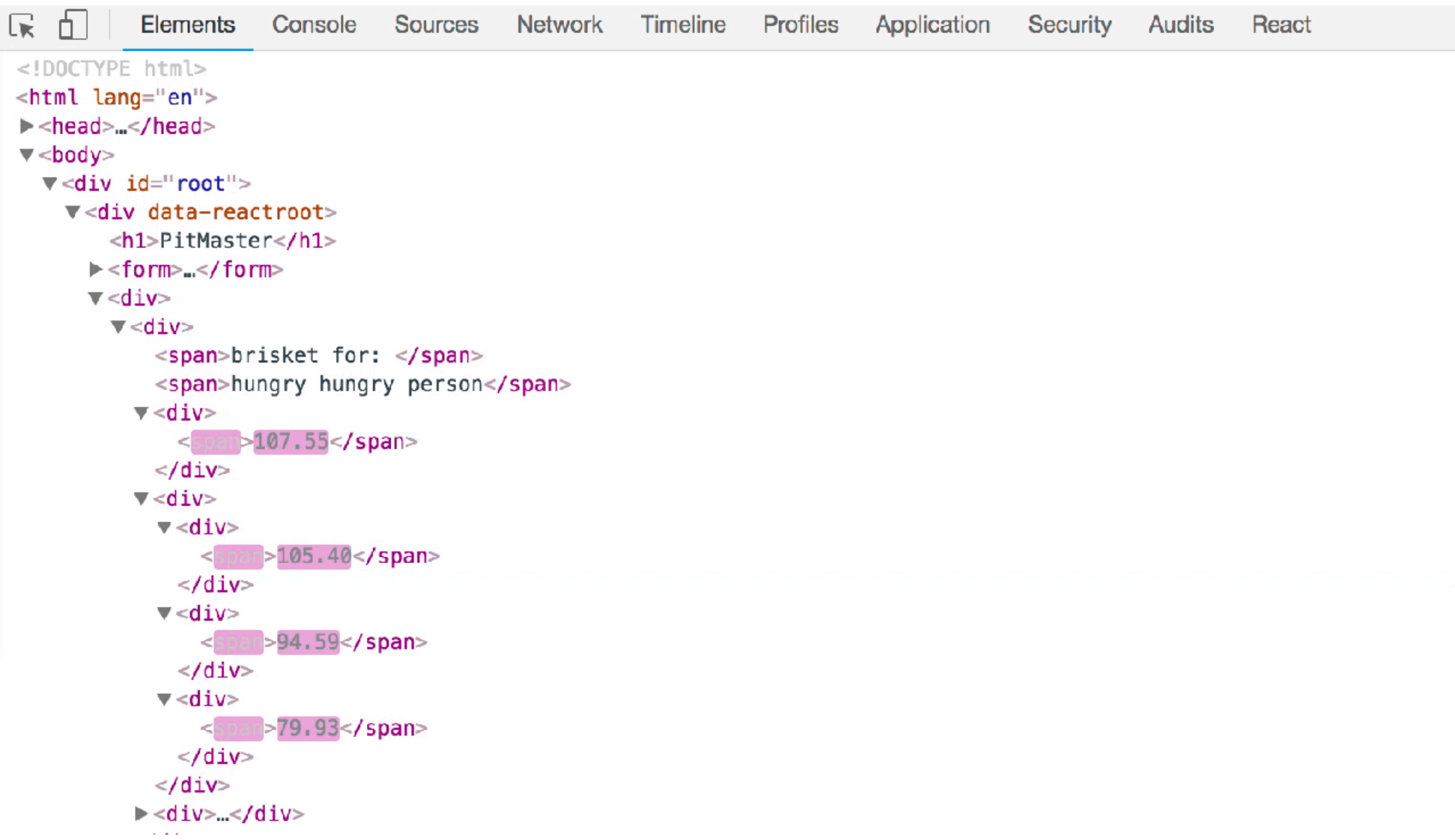
One-way data flow

Virtual DOM

- Plain object that represents the state of the DOM
- Results from nested calls to React.createElement
- Data can only come from props (arguments to React.createElement)
- New data (new arguments) cause new version of Virtual DOM to be calculated

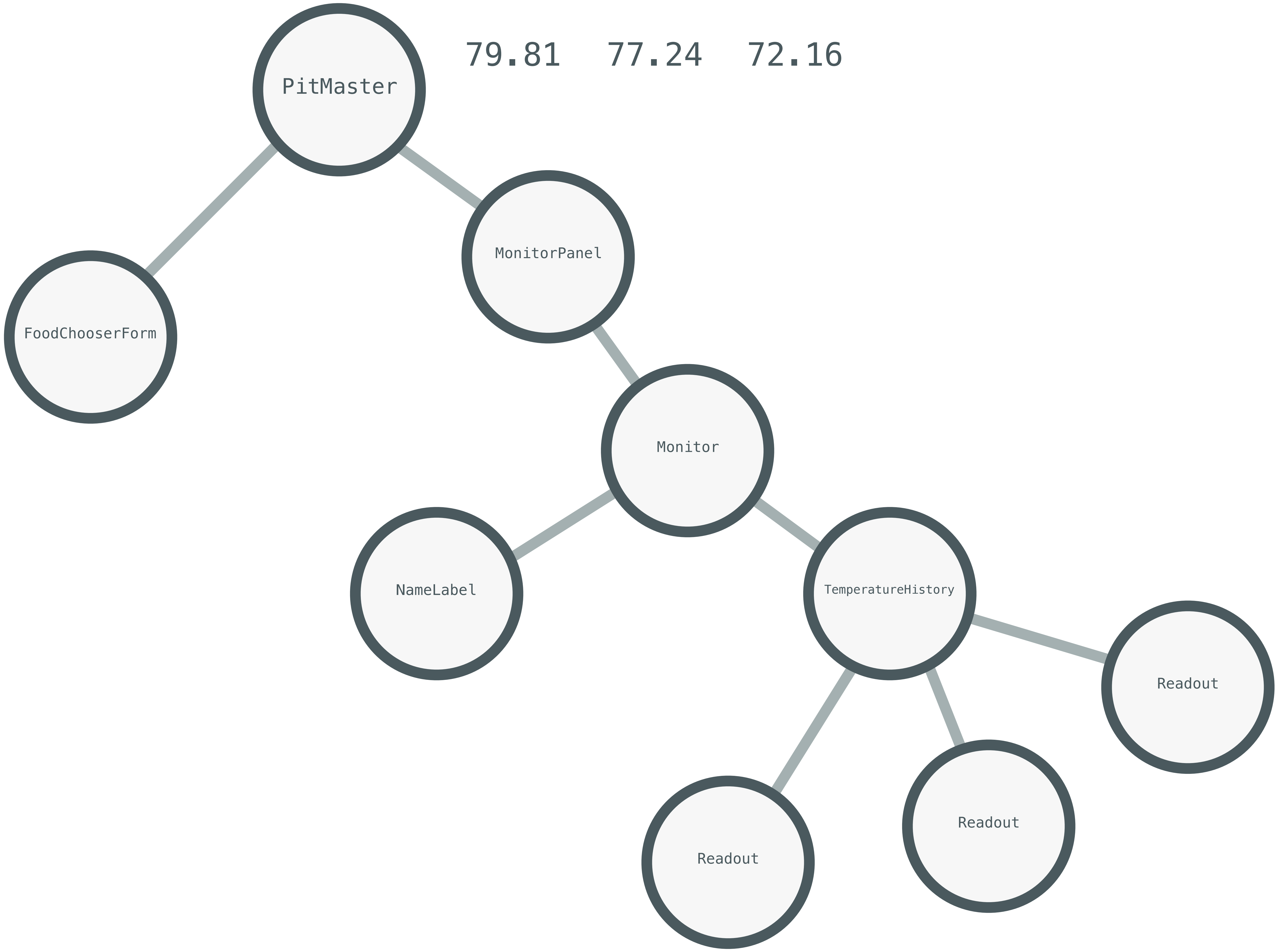

Elements
Console
Sources
Network
Timeline
Profiles
Application
Security
Audits
React
Trace React Updates
Highlight Search
Use Regular Expressions

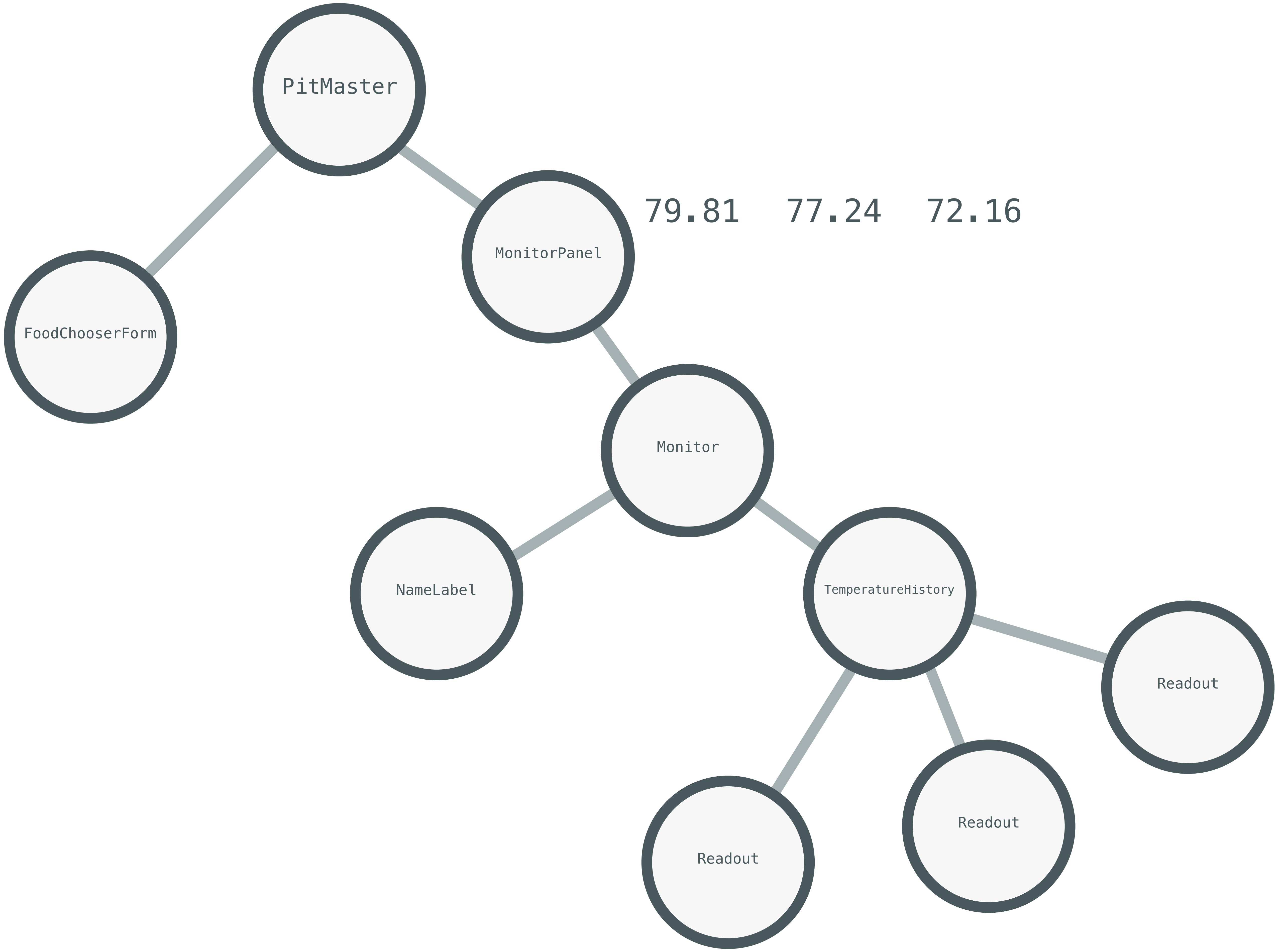
```
▼<PitMaster>
  ▼<div>
     <h1>PitMaster</h1>
   ▶<FoodChooserForm foodChoices=["brisket", "ribs", "wings", …] submitHandler=fn()>…</FoodChooserForm>
   ▼<MonitorPanel orderArray=[{…}]>
     ▼<div>
       ▼<Monitor key="1489958064410" name="hungry hungry person" food="brisket" foodTemperature=83.46624703801648…>
         ▼<div>
           ▶<NameLabel name="brisket for: ">…</NameLabel>
           ▶<NameLabel name="hungry hungry person">…</NameLabel>
           ▶<Readout value=83.46624703801648>…</Readout>
           ▼<TemperatureHistory valueArray=[80.94560036316938, 68.3051246754946, "--"]>
             ▼<div>
               ▶<Readout key="0" value=80.94560036316938>…</Readout>
               ▶<Readout key="1" value=68.3051246754946>…</Readout>
               ▶<Readout key="2" value="--">…</Readout>
               </div>
            </TemperatureHistory>
           ▶<Readout value=250>…</Readout>
          </div>
        </Monitor>
      </div>
    </MonitorPanel>
  </div>
</PitMaster>
```

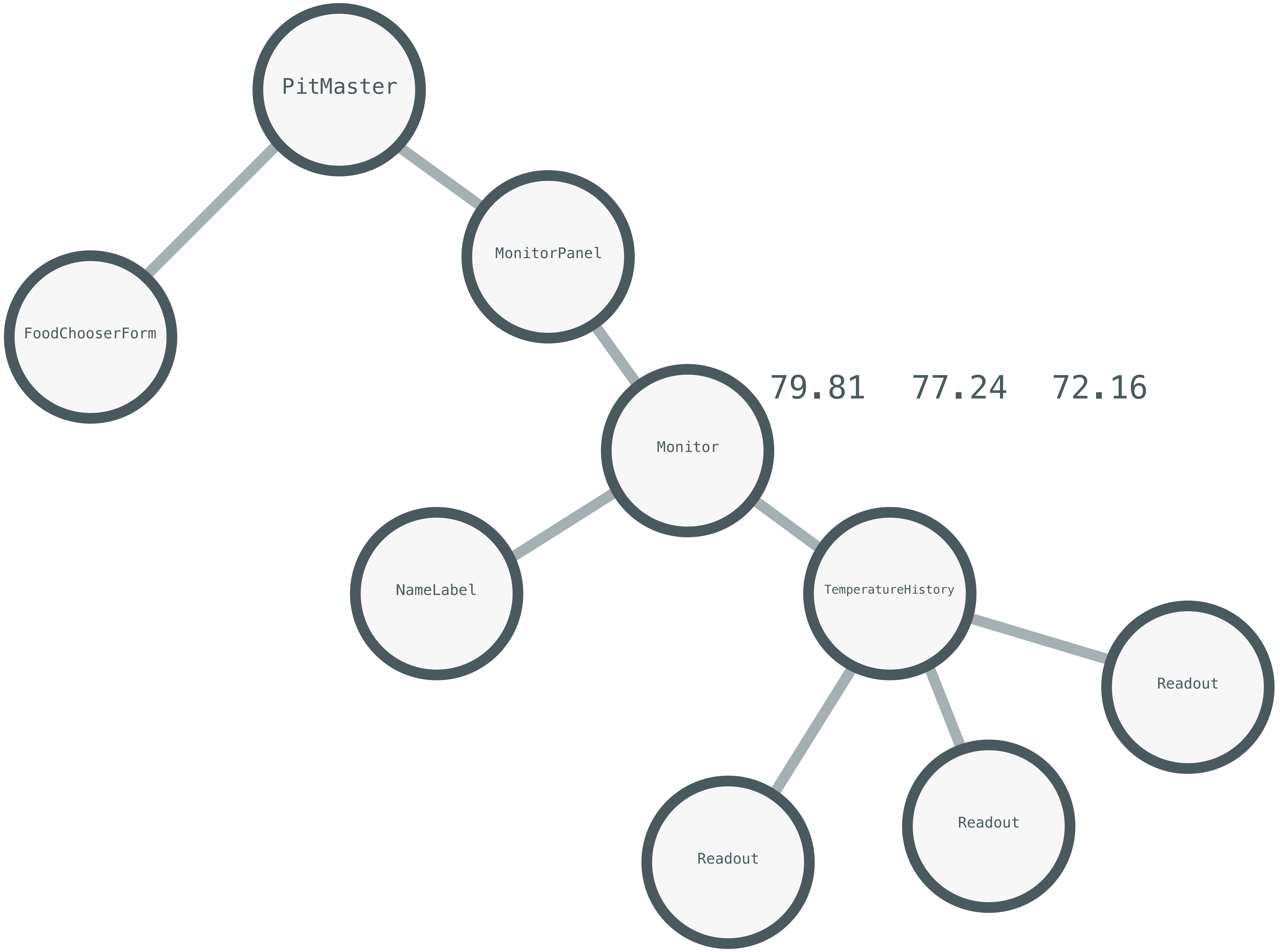


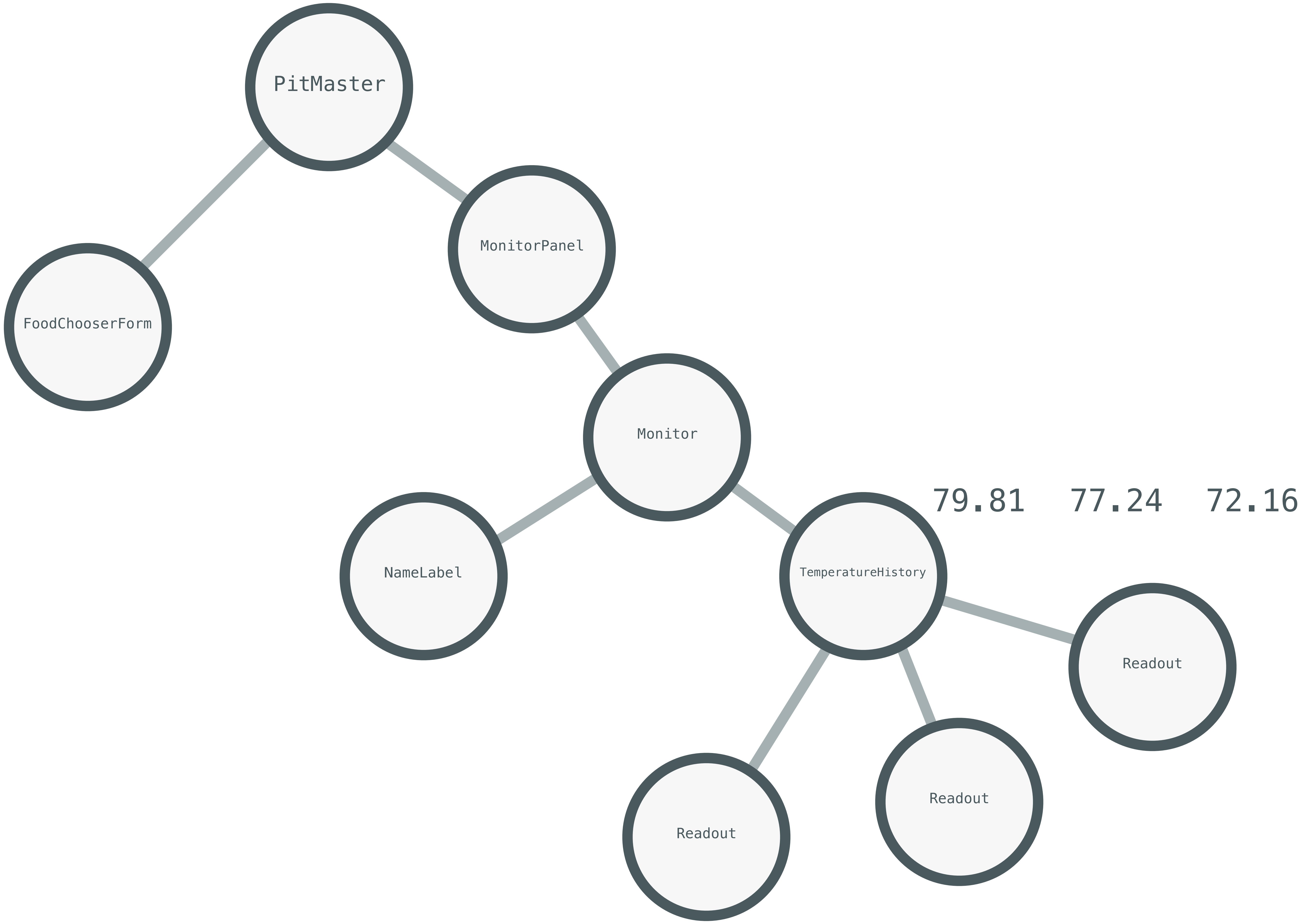
```
<!DOCTYPE html>
<html lang="en">
▶<head>…</head>
▼<body>
  ▼<div id="root">
    ▼<div data-reactroot>
        <h1>PitMaster</h1>
      ▶<form>…</form>
      ▼<div>
        ▼<div>
            <span>brisket for: </span>
            <span>hungry hungry person</span>
          ▼<div>
              <span>107.55</span>
            </div>
          ▼<div>
            ▼<div>
                <span>105.40</span>
              </div>
            ▼<div>
                <span>94.59</span>
              </div>
            ▼<div>
                <span>79.93</span>
              </div>
            </div>
          ▶<div>…</div>
```

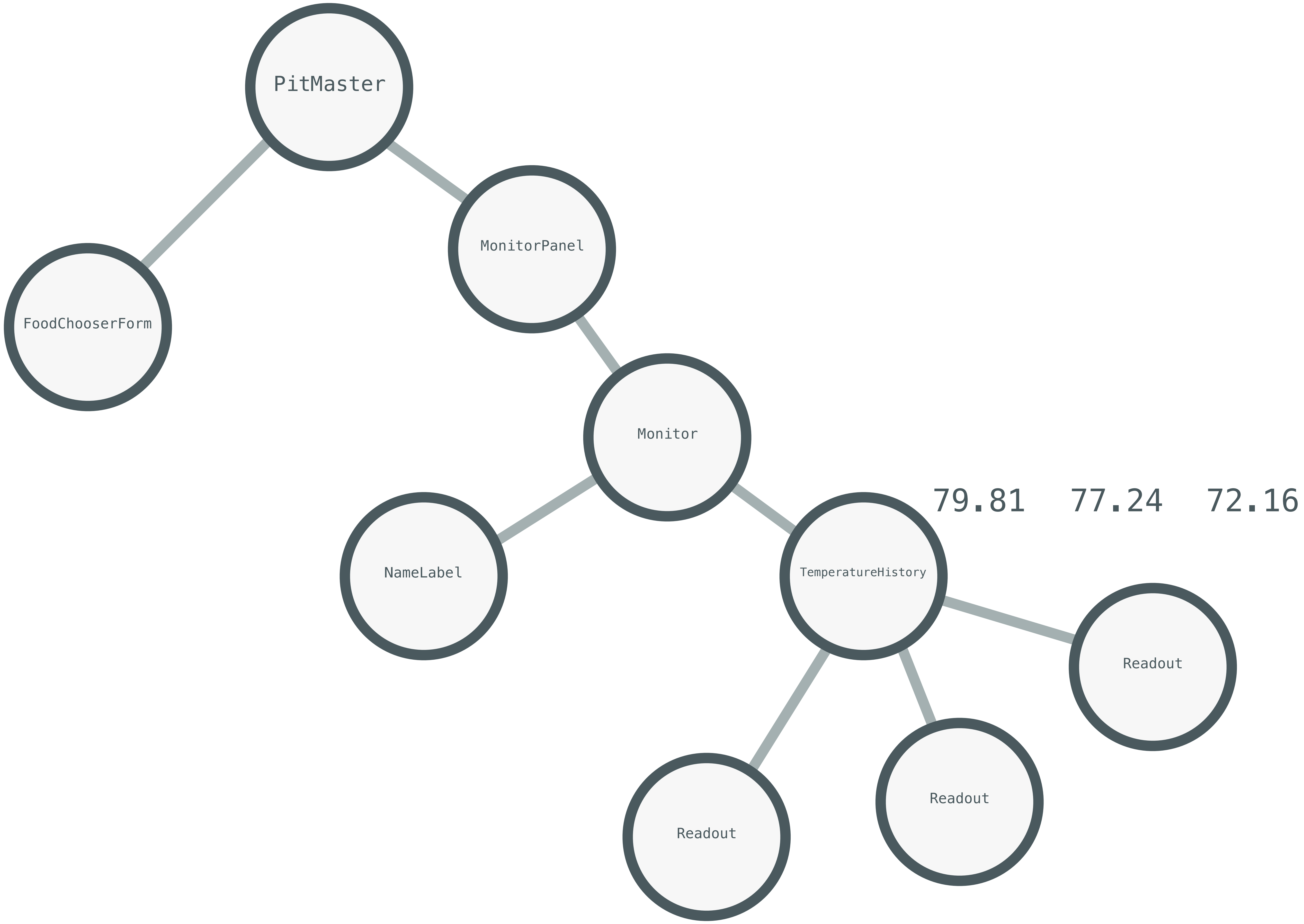
79.81   77.24   72.16

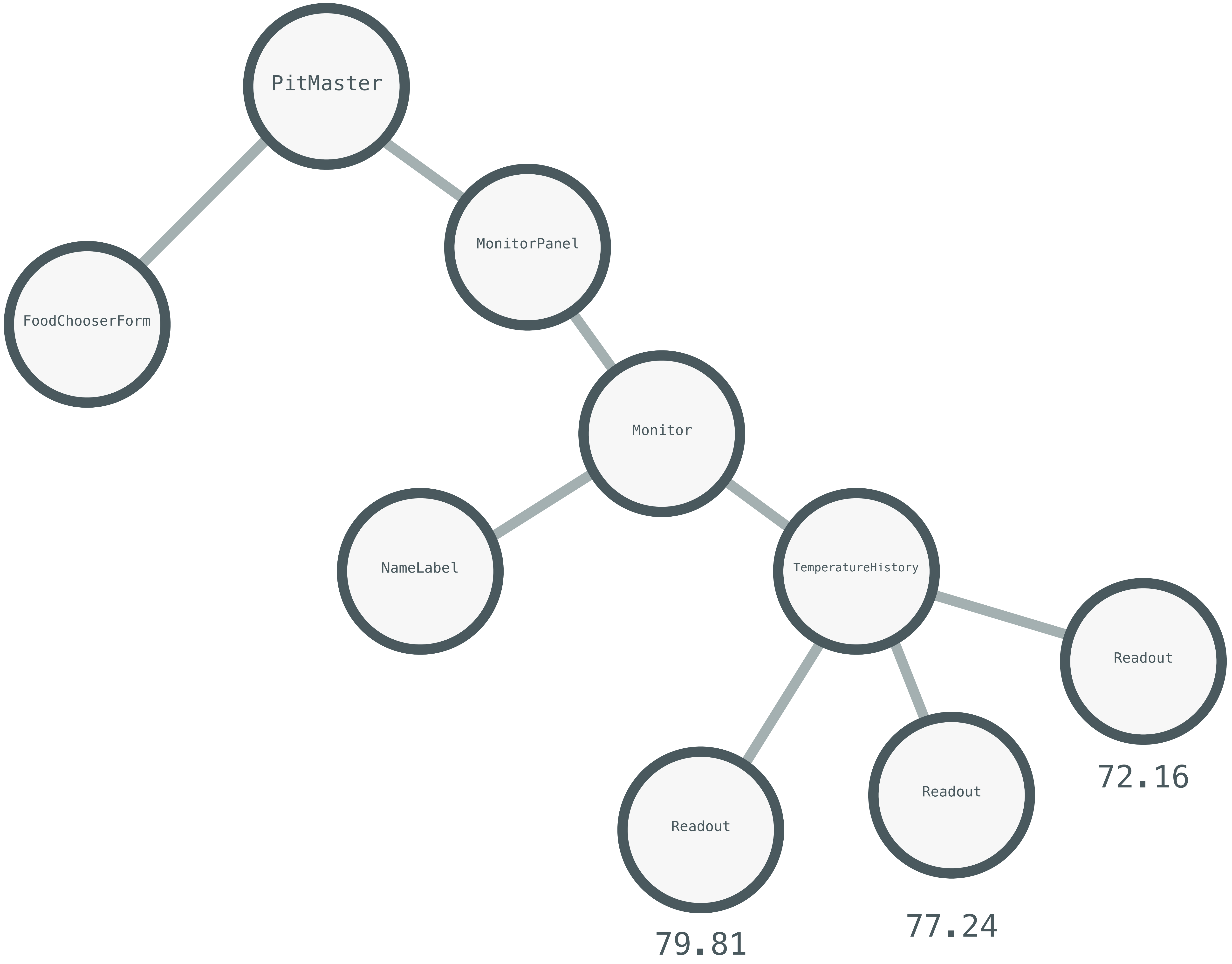


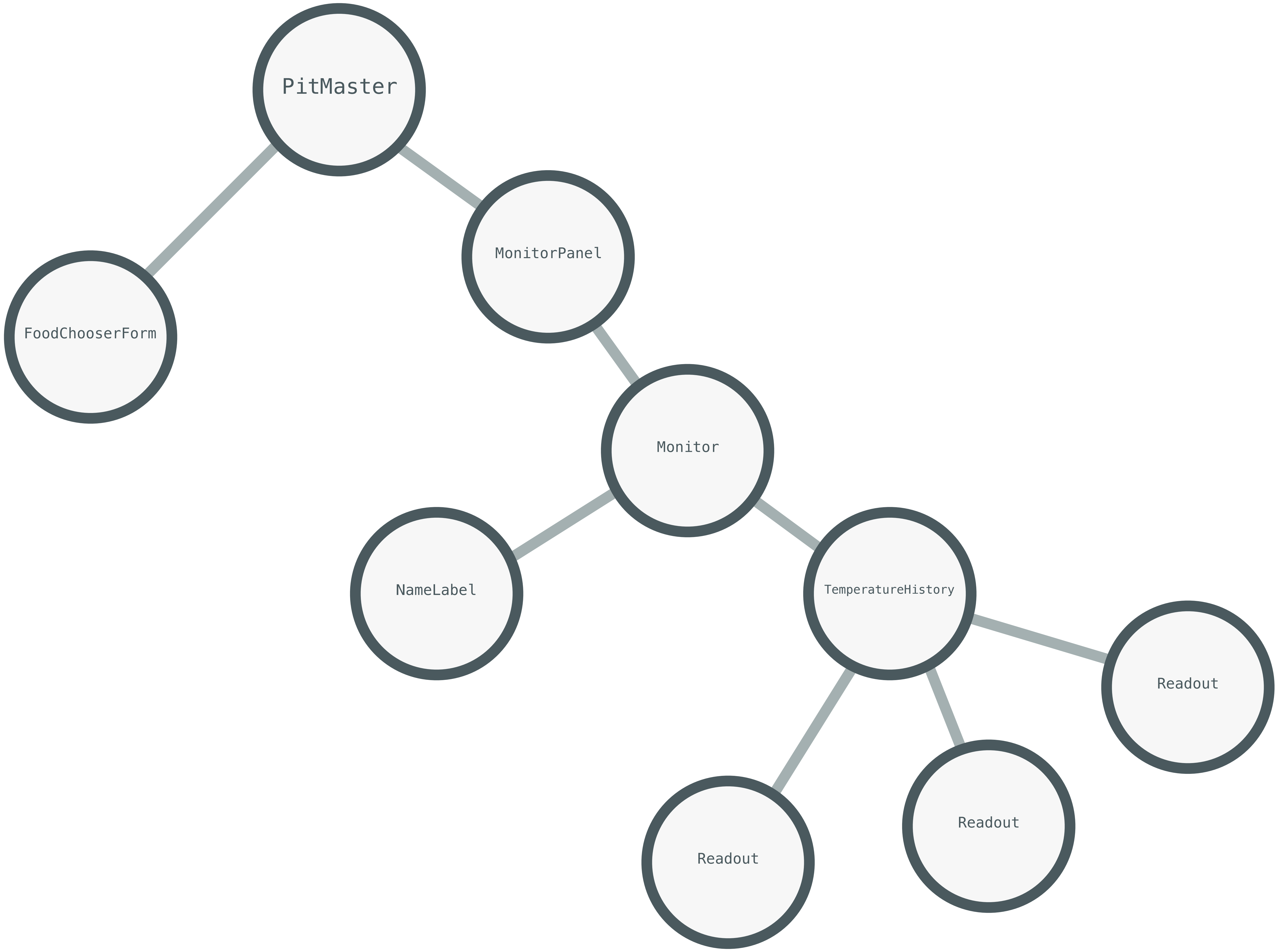

PitMaster
FoodChooserForm
MonitorPanel
79.81
77.24
72.16
Monitor
NameLabel
TemperatureHistory
Readout
Readout
Readout

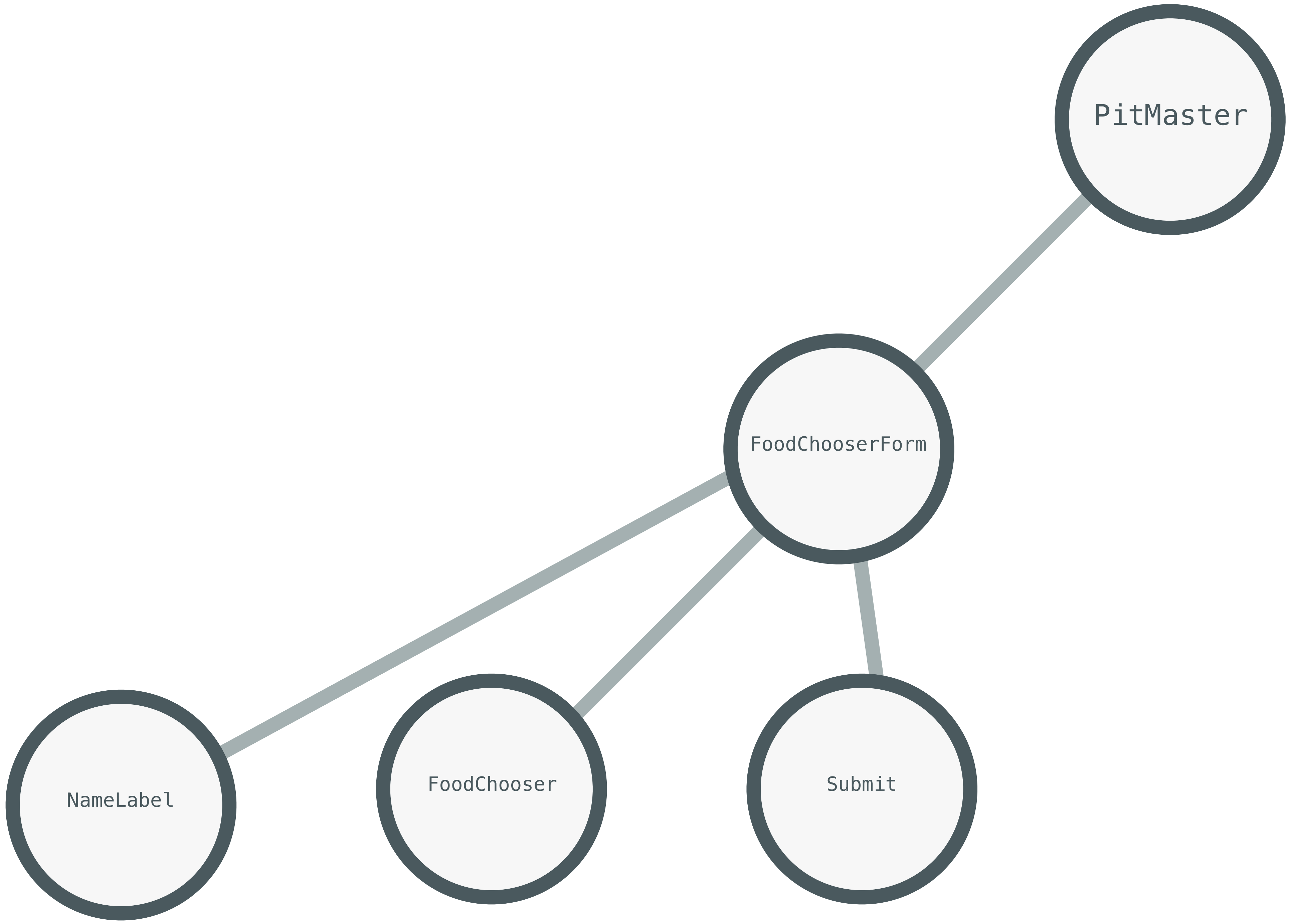

PitMaster
FoodChooserForm
MonitorPanel
Monitor
79.81
77.24
72.16
NameLabel
TemperatureHistory
Readout
Readout
Readout

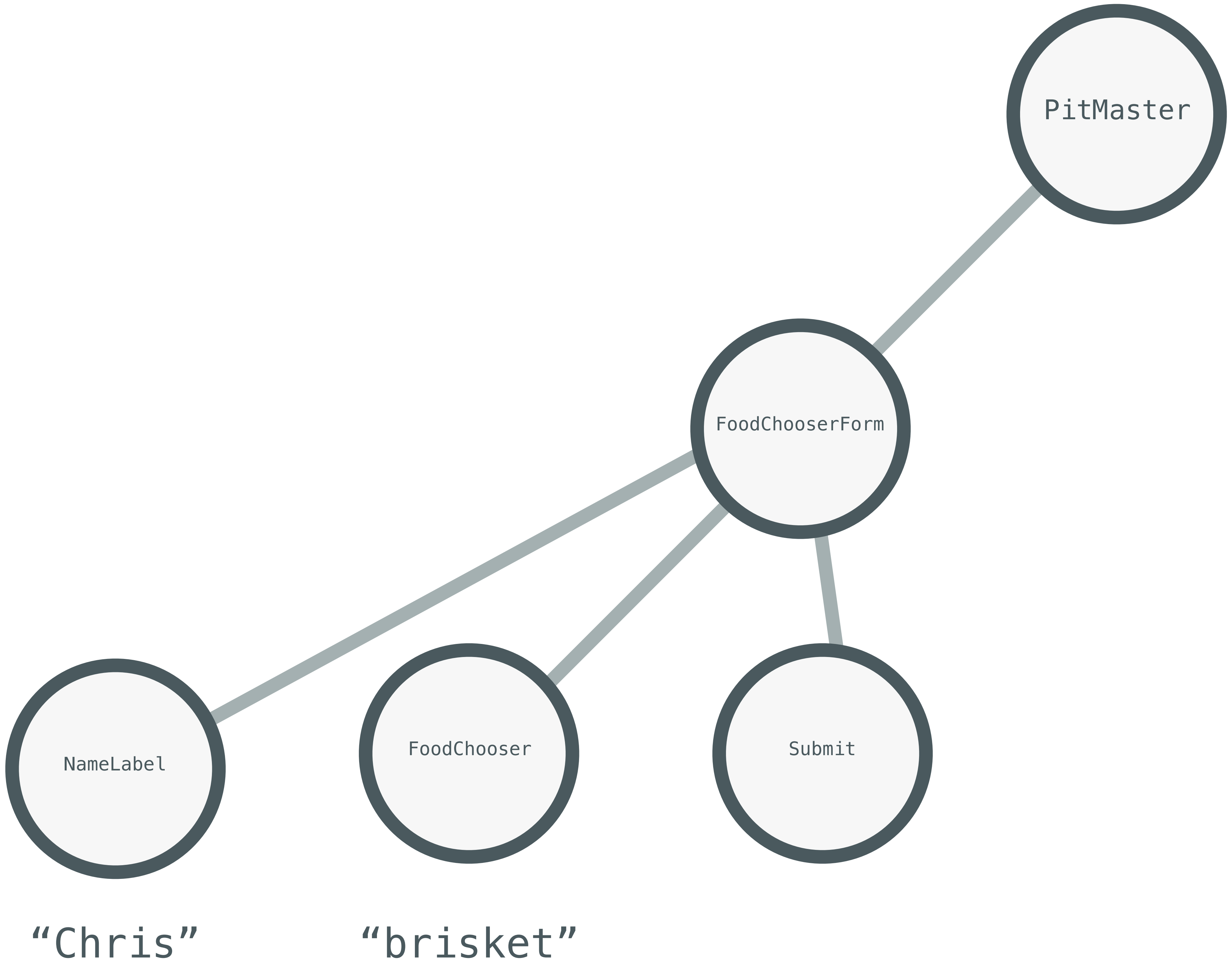

PitMaster
FoodChooserForm
MonitorPanel
Monitor
NameLabel
TemperatureHistory
79.81
77.24
72.16
Readout
Readout
Readout

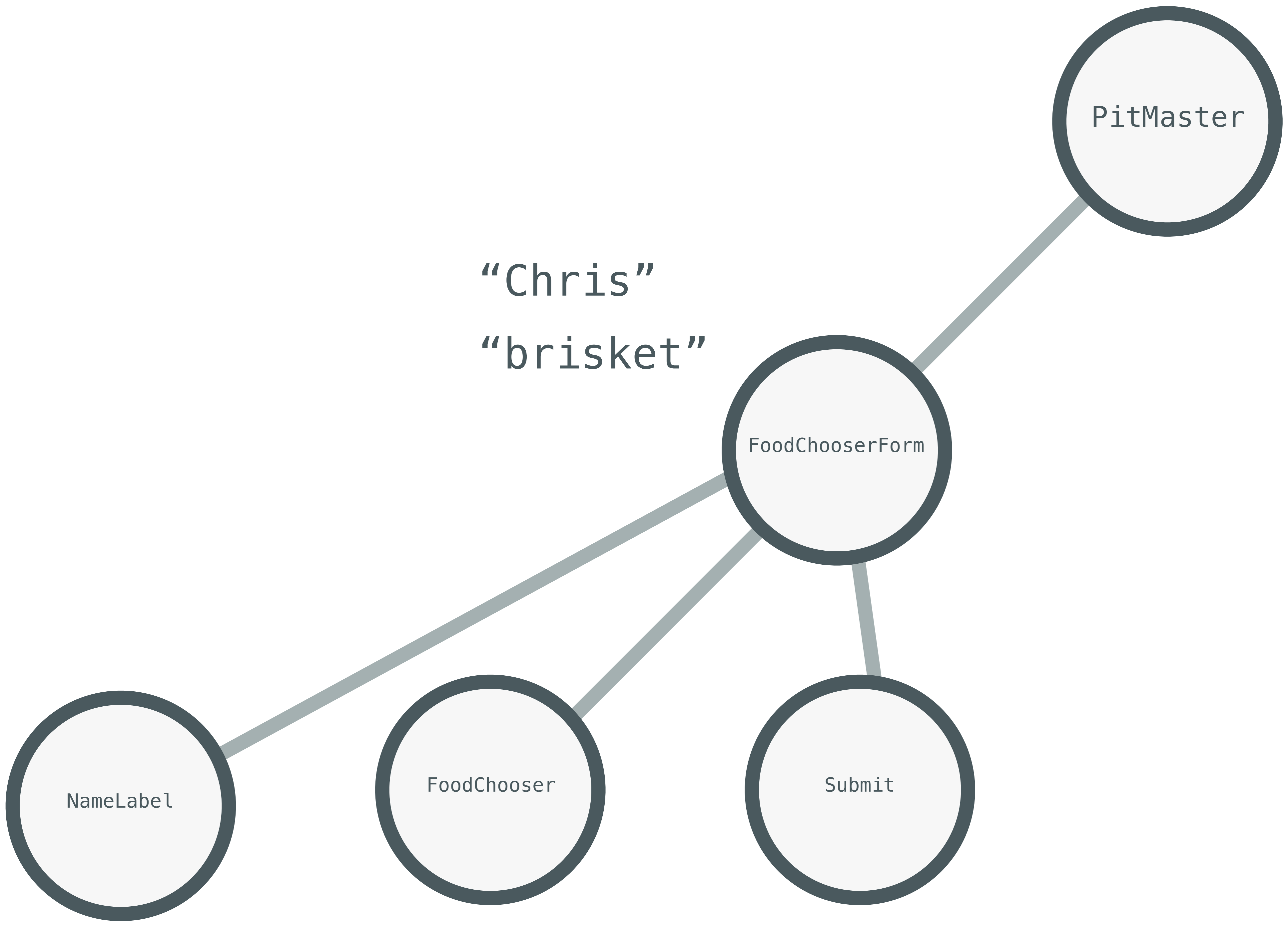

PitMaster
FoodChooserForm
MonitorPanel
Monitor
NameLabel
TemperatureHistory
79.81
77.24
72.16
Readout
Readout
Readout


PitMaster
FoodChooserForm
MonitorPanel
Monitor
NameLabel
TemperatureHistory
Readout
72.16
Readout
77.24
Readout
79.81

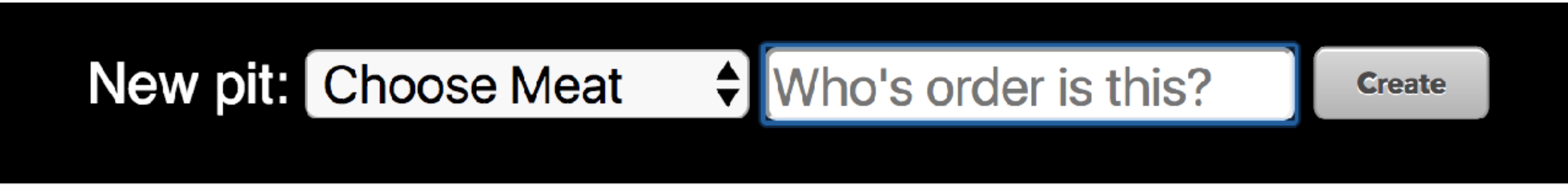

PitMaster
FoodChooserForm
MonitorPanel
Monitor
NameLabel
TemperatureHistory
Readout
Readout
Readout
Readout


PitMaster
FoodChooserForm
NameLabel
FoodChooser
Submit

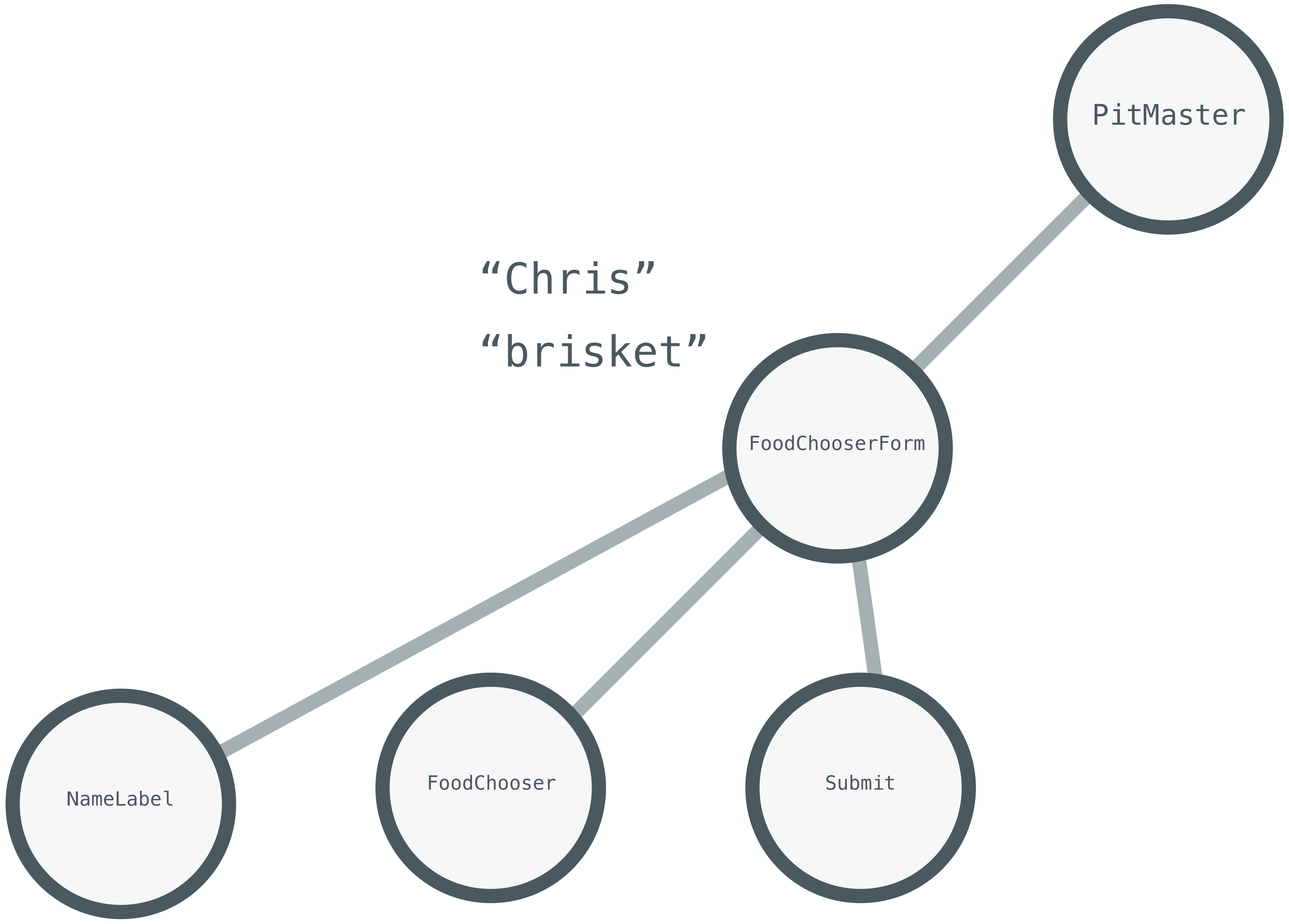

PitMaster
FoodChooserForm
NameLabel
FoodChooser
Submit
“Chris”
“brisket”


PitMaster
“Chris”
“brisket”
FoodChooserForm
NameLabel
FoodChooser
Submit

Classes


New pit:
Choose Meat
Who's order is this?
Create

```
const FoodChooserForm = () => (
  <form>
    <FoodChooser />
    <NameLabel />
    <input type="submit" />
  </form>
);

const NameLabel = ({name}) => (
  <input
    type="text"
    value={name}
  />
);
```


PitMaster
“Chris”
“brisket”
FoodChooserForm
NameLabel
FoodChooser
Submit

```
const NameLabel = ({name}) => (
  <input
    type="text"
    value={name}
  />
);
```

RETURN TO SENDER

```
const NameLabel = ({name, changeHandler}) => (
  <input
    type="text"
    value={name}
    onChange={changeHandler}
  />
);
```

Synthetic Events

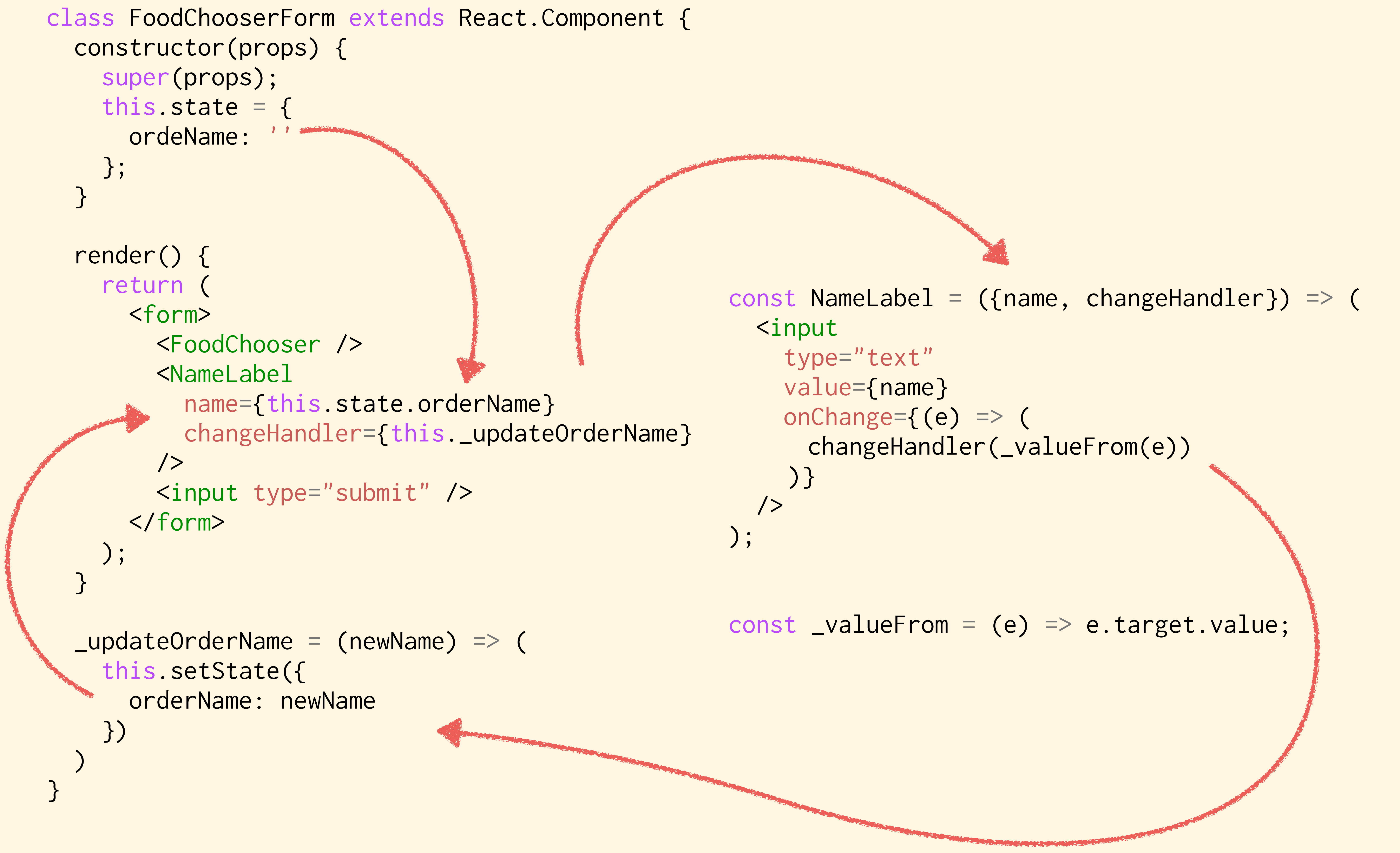
- Handlers for events are received as props
- Have names like "onClick", "onChange", "onSubmit"
- Allow user interaction to trigger handler functions

```
const NameLabel = ({name, changeHandler}) => (
  <input
    type="text"
    value={name}
    onChange={(e) => changeHandler(_valueFrom(e))}
  />
);


const _valueFrom = (e) => e.target.value;
const FoodChooserForm = () => (
  <form>
    <FoodChooser />
    <NameLabel
      name={/* what do i pass here? */}
      changeHandler={_updateOrderName} />
    <input type="submit" />
  </form>
);

const _updateOrderName = (val) => {
  // where do I store val?
};
```

```
class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
  }

  render() {
    return (
      <form>
        <FoodChooser />
        <NameLabel />
        <input type="submit" />
      </form>
    );
  }

  _updateOrderName = (val) => {

  }
}
```

```
class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form>
        <FoodChooser />
        <NameLabel />
        <input type="submit" />
      </form>
    );
  }

  _updateOrderName = (newName) => (
    this.setState({
      orderName: newName
    })
  )
}
```

```
class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }

  _updateOrderName = (newName) => (
    this.setState({
      orderName: newName
    })
  )
}
```



```
class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }

  _updateOrderName = (newName) => (
    this.setState({
      orderName: newName
    })
  )
}
```

```
const NameLabel = ({name, changeHandler}) => (
  <input
    type="text"
    value={name}
    onChange={(e) => (
      changeHandler(_valueFrom(e))
    )}
  />
);


const _valueFrom = (e) => e.target.value;
```

Controlled Components

- Values come only from props
- Update their values indirectly
- Are passed callback functions as props

```
class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }

  _updateOrderName = (newName) => (
    this.setState({
      orderName: newName
    })
  )
}
```

```
class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.submitHandler = props.submitHandler;
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form onSubmit={this.submitHandler}>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }

  _updateOrderName = (newName) => (
    this.setState({
      orderName: newName
    })
  )
}
```

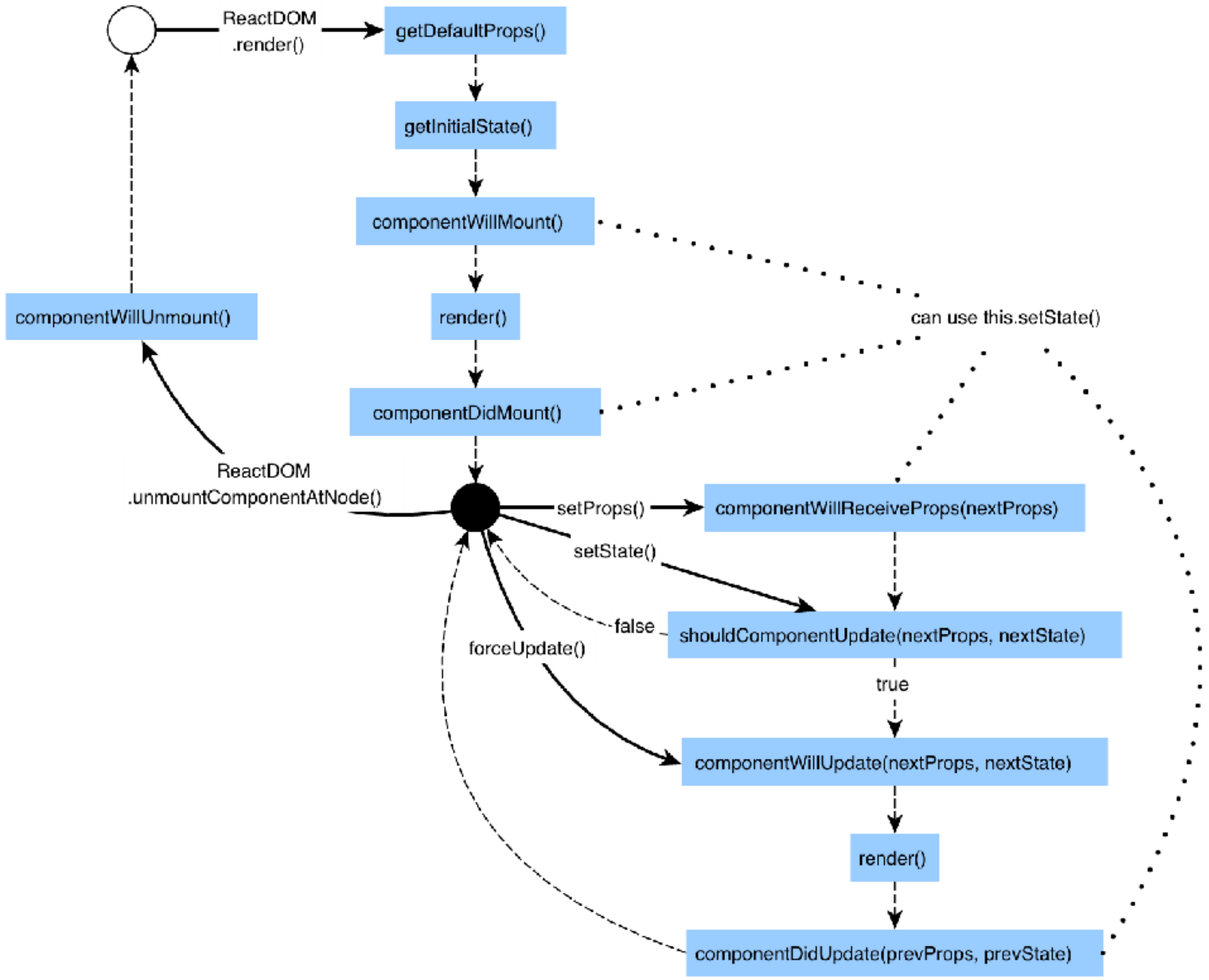
```
class PitMaster extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      orders: []
    };
  }
  render() {
    return (
      <div className="pitmaster">
        <h1>
          <img src={pitmasterLogo} alt="pitmaster" />
        </h1>
        <FoodChooserForm
          submitHandler={this._addOrder}
        />
      </div>
    );
  }
   _addOrder = (order) => {
     this.setState({
        orders: orders.concat[order]
     })
   }
}
```

```
class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.submitHandler = props.submitHandler;
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form onSubmit={this.submitHandler}>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }

  _updateOrderName = (newName) => (
    this.setState({
      orderName: newName
    })
  )
}
```

Component Classes

- When you need to save state between renders
- Define state change methods, pass methods as props
- State change methods call this.setState
- Changing state causes re-render
- Extend React.Component

```
class React.Component {
  // mounting
  constructor(props) { /* ... */}
  componentWillMount() { /* ... */}
  render() { /* ... */}
  componentDidMount() { /* ... */}

  // updating
  componentWillReceiveProps() { /* ... */}
  shouldComponentUpdate() { /* ... */}
  componentWillUpdate() { /* ... */}
  componentDidUpdate() { /* ... */}

  // unmount
  componentWillUnmount() { /* ... */}

  // misc
  setState() { /* ... */}
  forceUpdate() { /* ... */}
}
```


ReactDOM
.render()
getDefaultProps()
getInitialState()
componentWillMount()
render()
componentDidMount()
componentWillUnmount()
can use this.setState()
ReactDOM
.unmountComponentAtNode()
setProps()
componentWillReceiveProps(nextProps)
setState()
false
shouldComponentUpdate(nextProps, nextState)
forceUpdate()
true
componentWillUpdate(nextProps, nextState)
render()
componentDidUpdate(prevProps, prevState)

Component Classes

- Can hold and change state
- Has lifecycle methods automatically called by React
- Used sparingly!

Modules

```
class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.submitHandler = props.submitHandler;
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form onSubmit={this.submitHandler}>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }
 /* omitted */
}
```

```
const NameLabel = ({name, changeHandler}) => (
  <input
    type="text"
    value={name}
    onChange={changeHandler}
  />
);
```

```
class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.submitHandler = props.submitHandler;
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form onSubmit={this.submitHandler}>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }
}
```

```
const NameLabel = ({name, changeHandler}) => (
  <input
    type="text"
    value={name}
    onChange={changeHandler}
  />
);
```

```
const NameLabel = ({name, changeHandler}) => (
  <input
    type="text"
    value={name}
    onChange={changeHandler}
  />
);

export default NameLabel;
```

```
const NameLabel = ({name, changeHandler}) => (
  <input
    type="text"
    value={name}
    onChange={changeHandler}
  />
);

export default NameLabel;
```

```
import React from 'react';
import FoodChooser from '../containers/FoodChooser';
import NameLabel from '../containers/NameLabel';

class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.submitHandler = props.submitHandler;
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form onSubmit={this.submitHandler}>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }
}
```

```
import React from 'react';
import FoodChooser from '../containers/FoodChooser';
import NameLabel from '../containers/NameLabel';

class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.submitHandler = props.submitHandler;
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form onSubmit={this.submitHandler}>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }
}
```

```
import React from 'react';
import FoodChooser from '../containers/FoodChooser';
import NameLabel from '../containers/NameLabel';

class FoodChooserForm extends React.Component {
  constructor(props) {
    super(props);
    this.submitHandler = props.submitHandler;
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form onSubmit={this.submitHandler}>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }
}
```

```
import {Component} from 'react';
import FoodChooser from '../containers/FoodChooser';
import NameLabel from '../containers/NameLabel';

class FoodChooserForm extends Component {
  constructor(props) {
    super(props);
    this.submitHandler = props.submitHandler;
    this.state = {
      ordeName: ''
    };
  }

  render() {
    return (
      <form onSubmit={this.submitHandler}>
        <FoodChooser />
        <NameLabel
          name={this.state.orderName}
          changeHandler={this._updateOrderName}
        />
        <input type="submit" />
      </form>
    );
  }
}
```

```
export {
  cookFood:cookFood,
  Sensor: Sensor
}
```

```
import {
  Sensor
} from '../lib/GrillSimulator';
```

```
export {
  cookFood,
  Sensor
}
```

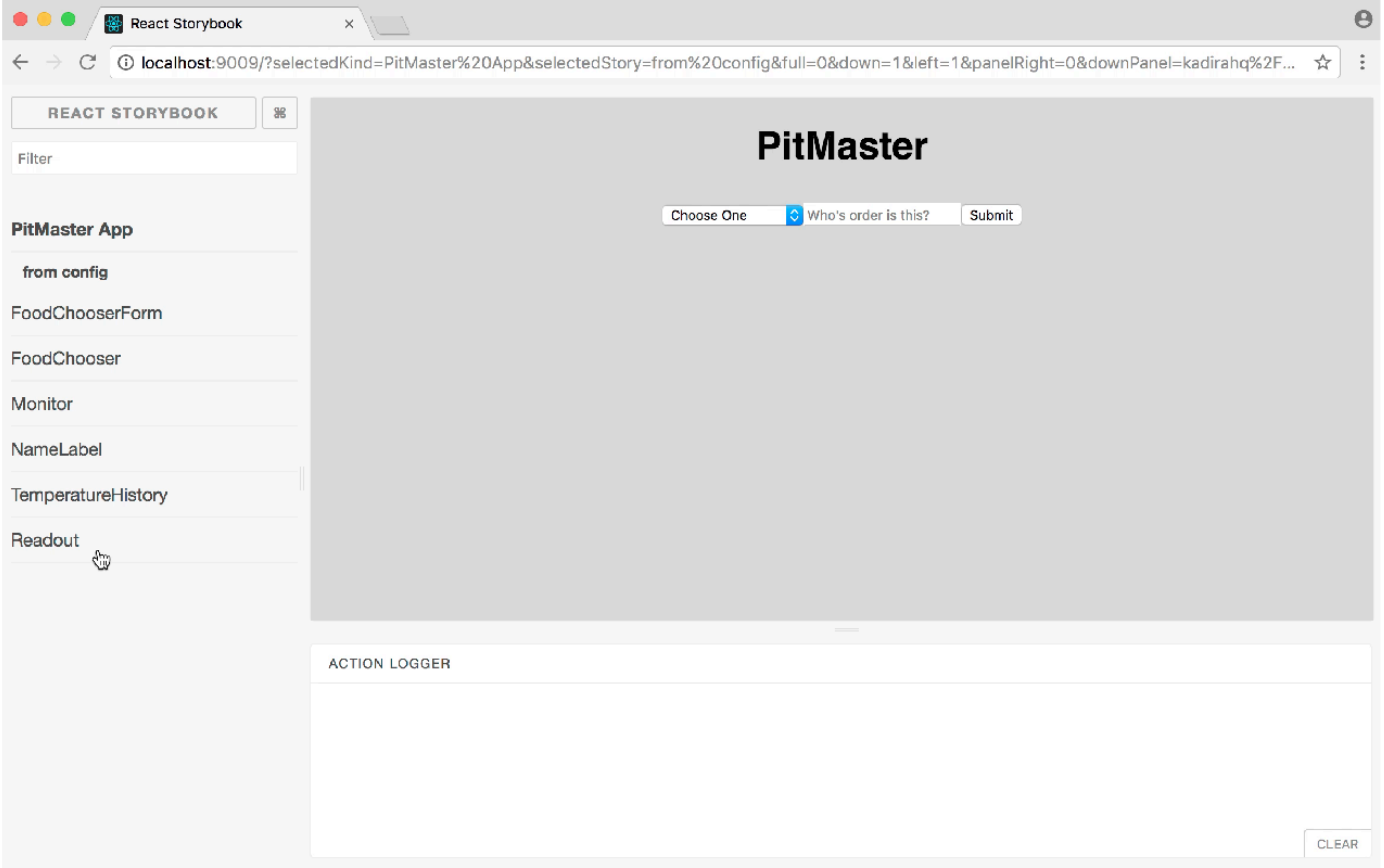
```
import {
  Sensor
} from '../lib/GrillSimulator';
```

Module syntax

- Keep your components organized
- One component per .js file
- Keep assorted helper functions in their own file
- Export using enhanced object literal syntax

```jsx
import React from 'react';
import styles from './Readout.css';

const Readout = ({value=0}) => (
  <div>
    <span className={styles.large} >{
      typeof value === 'number' ? value.toFixed(2)
                                : value
      }</span>
  </div>
);

export default Readout;
```

```css
.large {
  font-size: 18px;
}
.medium {
  font-size: 14px;
}
.small {
  font-size: 12px;
}
.smallest {
  font-size: 10px;
}
```


React Storybook
localhost:9009/?selectedKind=PitMaster%20App&selectedStory=from%20config&full=0&down=1&left=1&panelRight=0&downPanel=kadirahq%2F...
REACT STORYBOOK
Filter
PitMaster App
from config
FoodChooserForm
FoodChooser
Monitor
NameLabel
TemperatureHistory
Readout
PitMaster
Choose One
Who's order is this?
Submit
ACTION LOGGER
CLEAR

CSS Modules

- Use short, descriptive class names
- Scopes styles to components
- Class names are hashed in both bundled css and js files
- **Not yet supported in create-react-app**

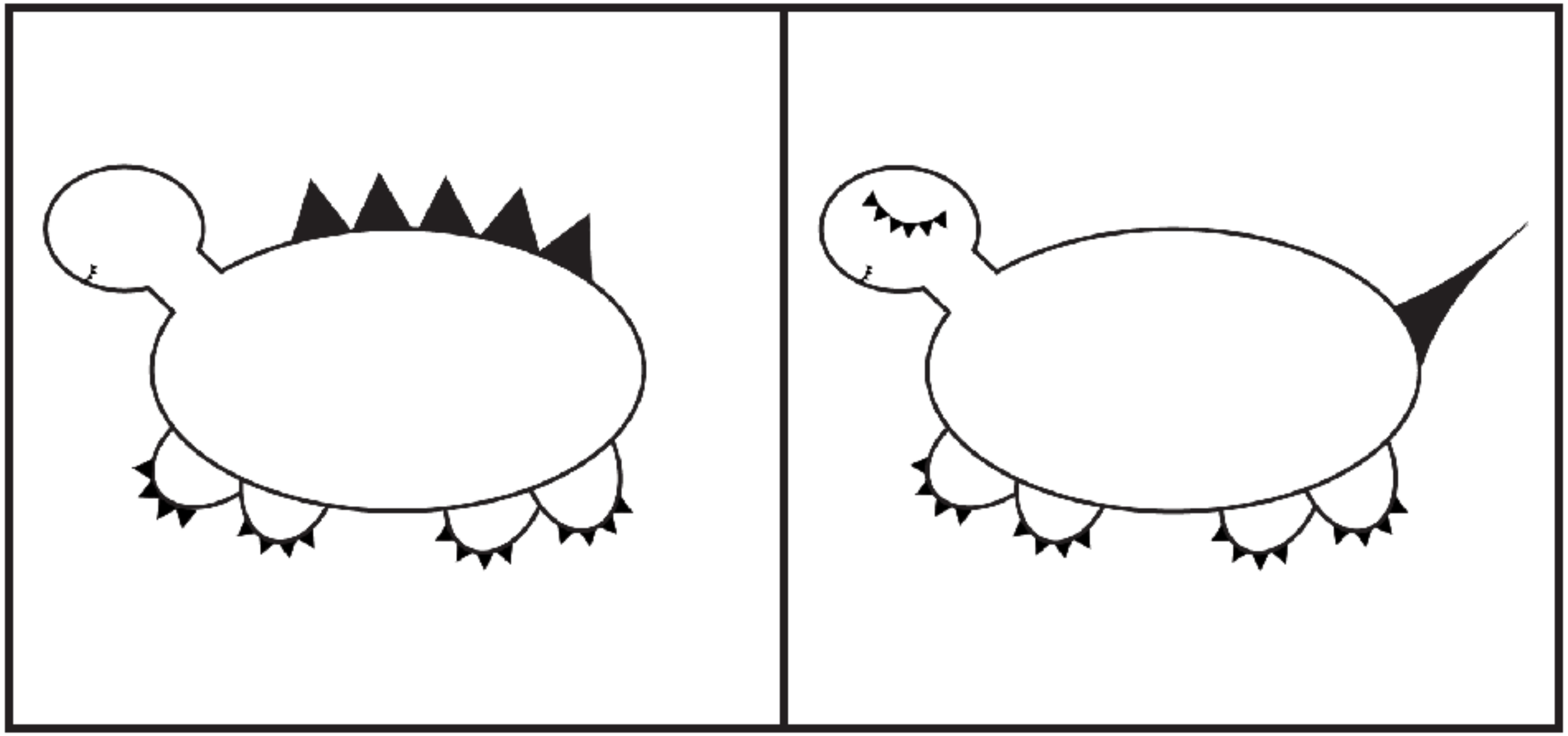
Modules

- Code organization is much, much nicer with build tools.
- create-react-app provides a convenient pre-configured React environment
- Be on the look out for support for CSS modules

Immutability

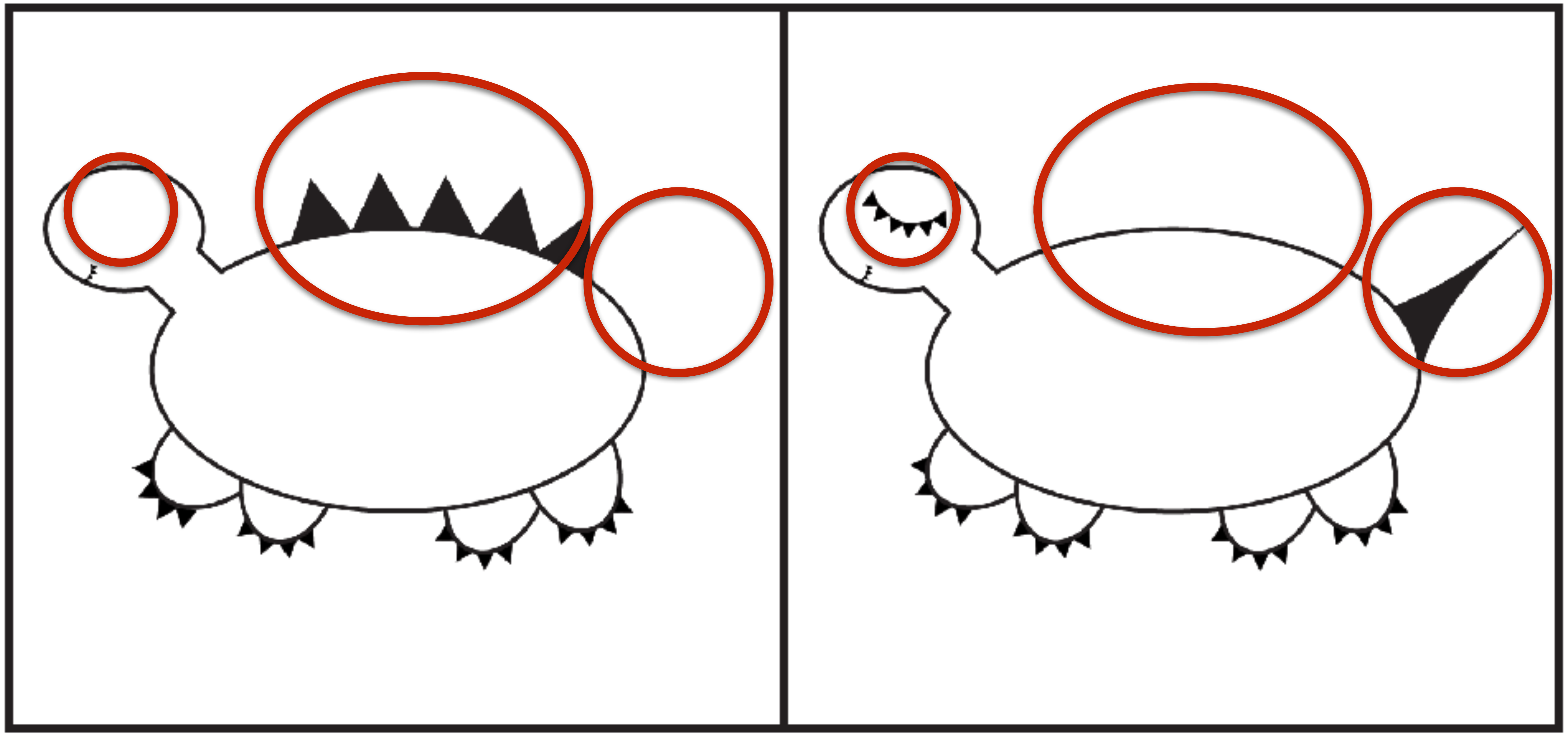
Spot the difference

Find 3 differences.

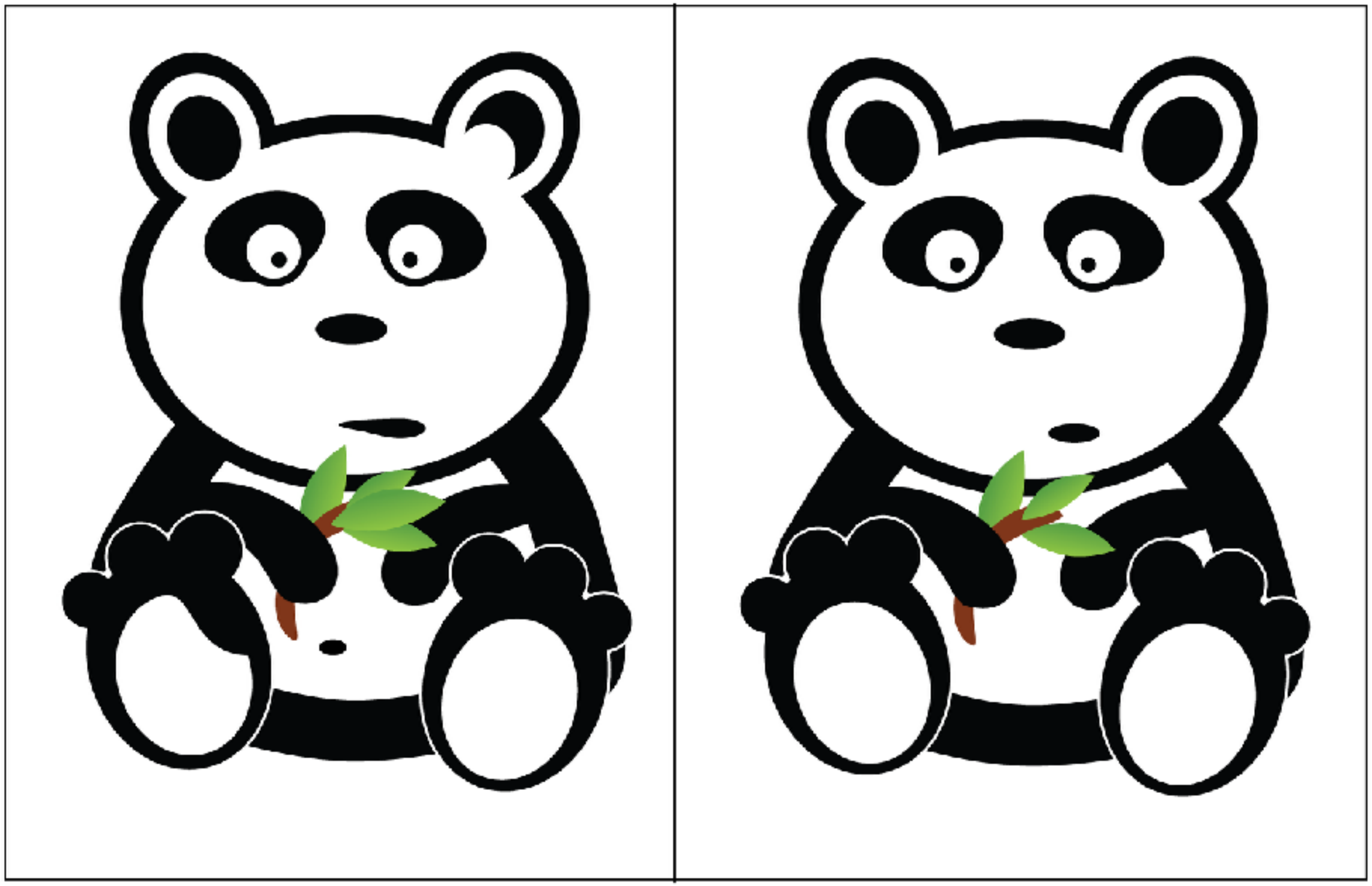


Spot the difference

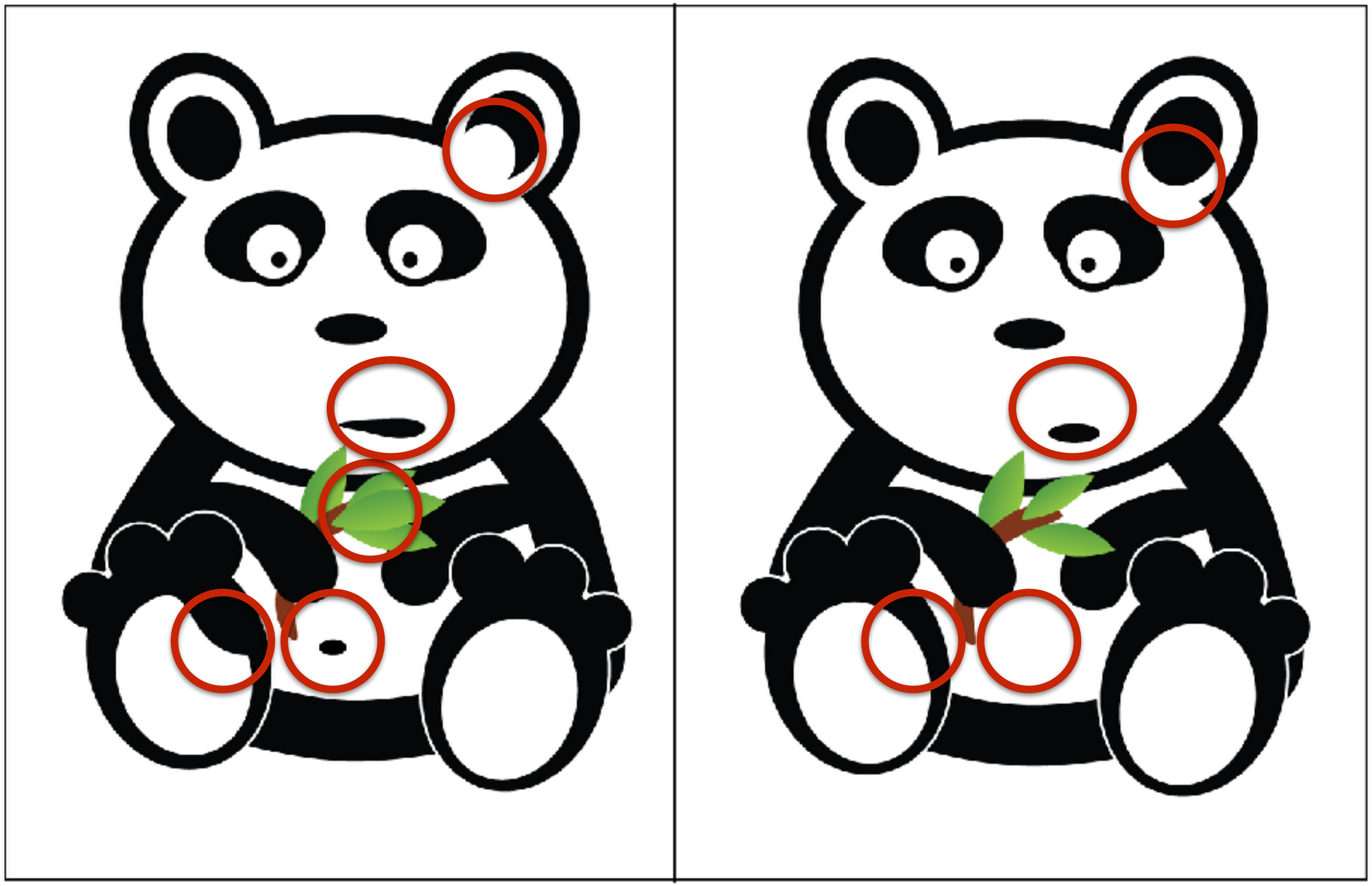
Find 3 differences.

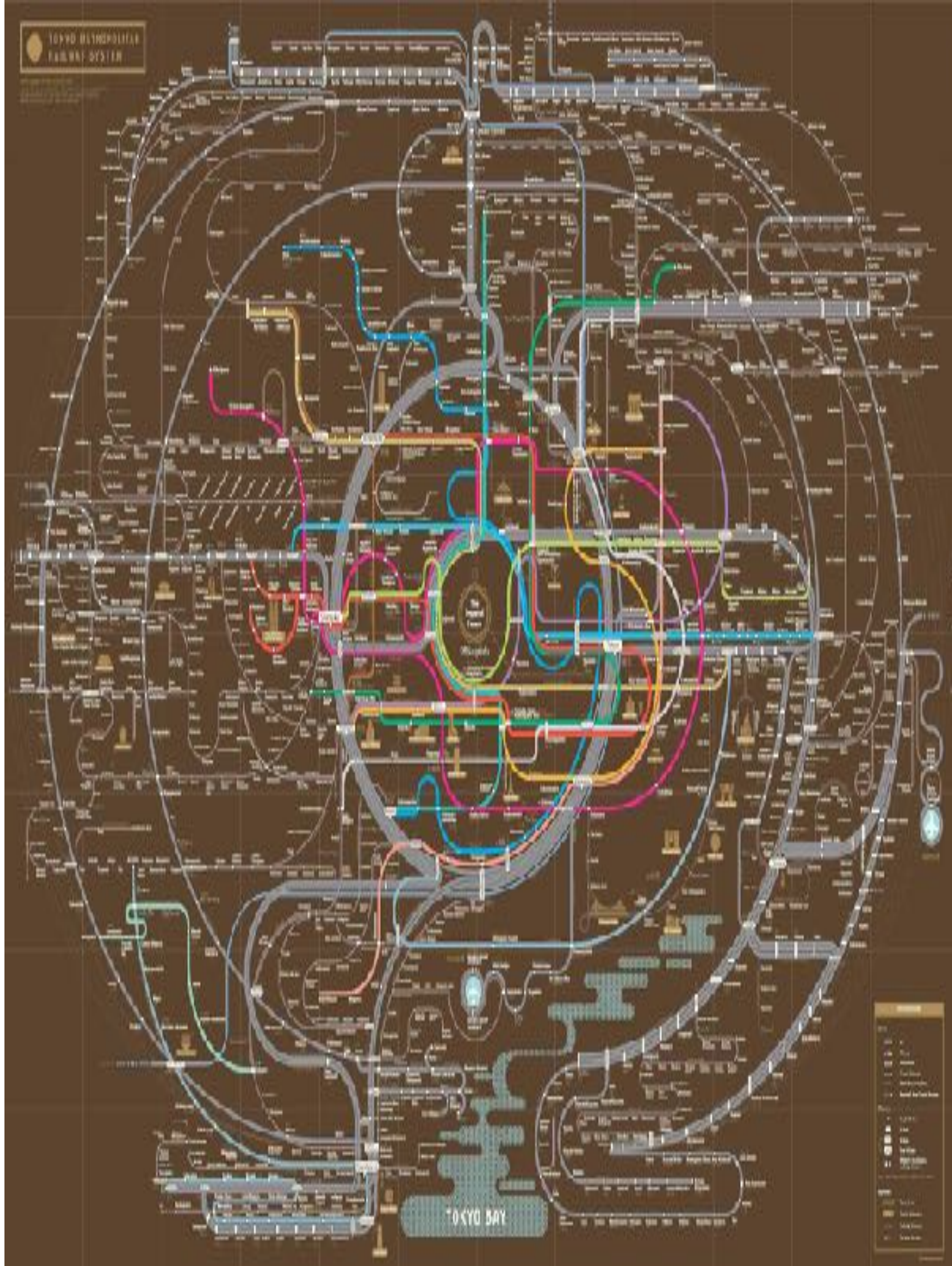


Spot the Five Differences

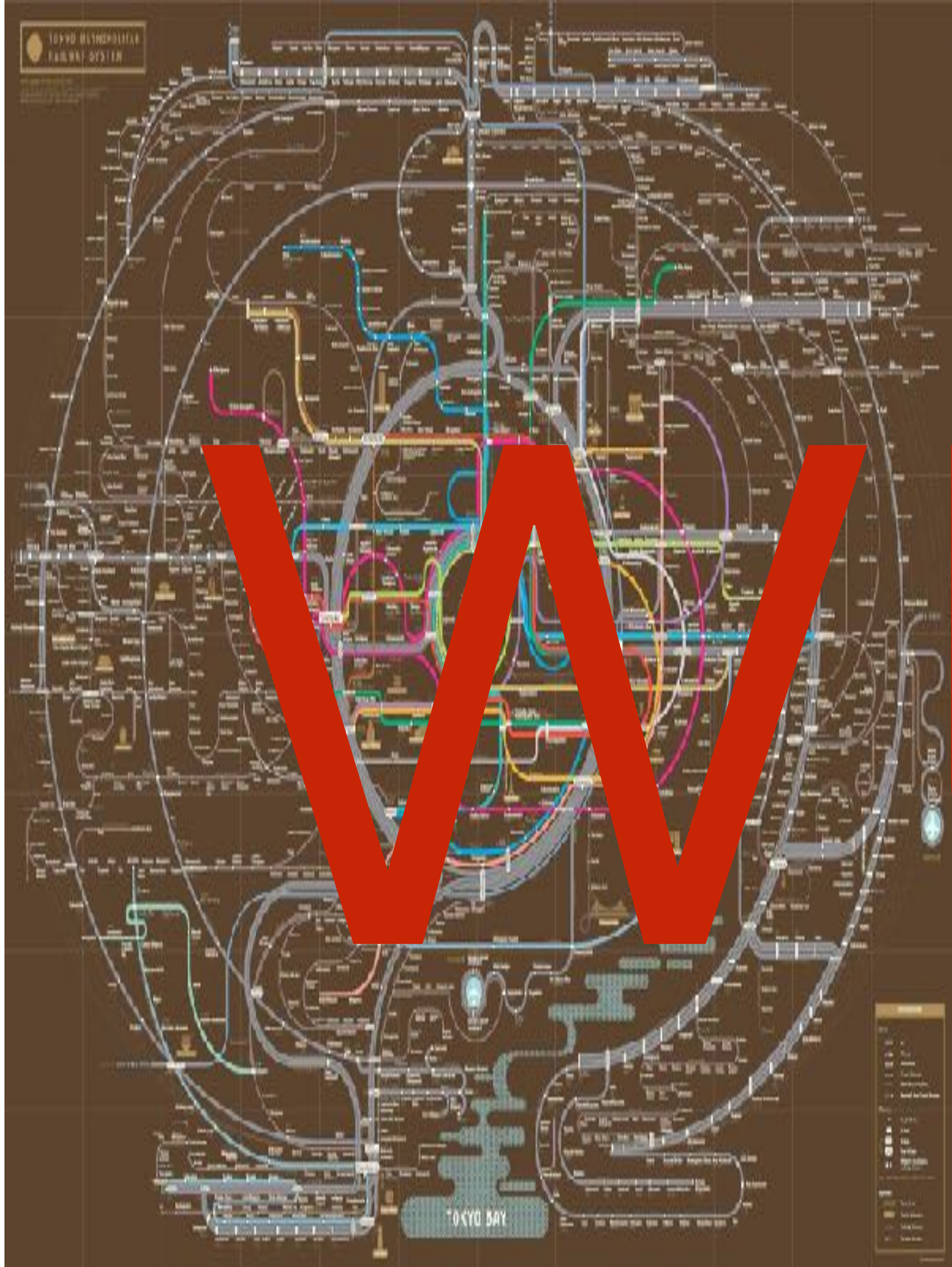


Spot the Five Differences




TOKYO METROPOLITAN
RAILWAY SYSTEM
The Imperial Palace
TOKYO BAY

TOKYO METROPOLITAN
RAILWAY SYSTEM
The Imperial Palace
TOKYO BAY


TOKYO METROPOLITAN RAILWAY SYSTEM
TOKYO BAY

TOKYO METROPOLITAN RAILWAY SYSTEM
TOKYO BAY

wut

Immutability + Performance

- componentShouldUpdate()
- Immutable.js

Object.assign

- Like jQuery’s .extend
- Returns single object with properties of multiple objects
- Rightmost object takes precedence

```
_updateTemperatures = (id) => {
   this.setState({
     orders: this.state.orders.map((order) => (
       order.id === id ? Object.assign({},
                           order,
                           {current: order.sensor.current()}
                         )}
                       : order

     ))
   })
 }
```

Array.concat

- Merges elements of two or more arrays
- Returns new array

```
_addOrder = (order) => {
   order.id = (new Date()).getTime();
   order.sensor = new Sensor(cookFood(ROOM_TEMP,
                                      TARGET_TEMP,
                                      FOOD_TYPE));
   this.setState({
     orders: this.state.orders.concat([order])
   });
   order.sensor.start();
 }
```

Object spread

- Like Object.assign
- Returns new object
- Rightmost properties take precedence

```
_updateTemperatures = (id) => {
   this.setState({
     orders: this.state.orders.map((order) => (
       order.id === id ? {
                           ...order,
                           current: order.sensor.current()
                         }
                       : order

     ))
   })
 }
```

Array spread

- Like Array.concat
- Merges arrays and elements
- Returns new array

```
_addOrder = (order) => {
   order.id = (new Date()).getTime();
   order.sensor = new Sensor(cookFood(ROOM_TEMP,
                                      TARGET_TEMP,
                                      FOOD_TYPE));
   this.setState({
     orders: [...this.state.orders, order]
   });
   order.sensor.start();
 }
```

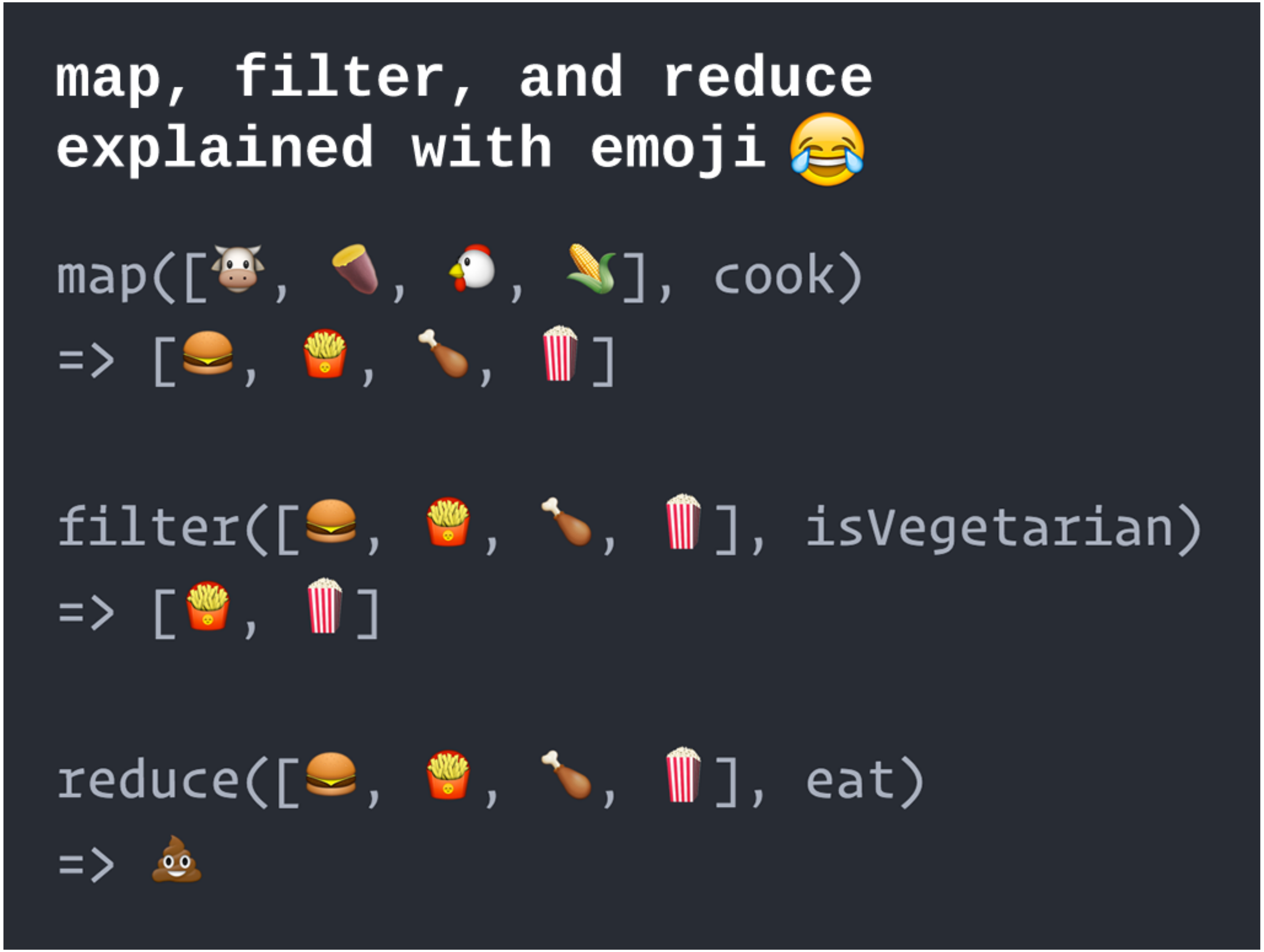
```
_addOrder = (order) => {
   order.id = (new Date()).getTime();
   order.sensor = new Sensor(cookFood(ROOM_TEMP,
                                      TARGET_TEMP,
                                      FOOD_TYPE));
   this.setState({
     orders: [order, ...this.state.orders]
   });
   order.sensor.start();
 }
```

Functional Array methods

- Map - transforms every value
- Reduce - returns a single value
- Filter - returns values that pass criteria

```
_removeOrder = (id) => (
    this.setState({
      orders: this.state.orders.filter((order) => order.id === id)
    })
 )
```

```
_totalCurrentTemperature = () => (
   this.state.orders.reduce((runningTotal, {current}) => (
    runningTotal + current
   ), 0)
 )
```



map, filter, and reduce explained with emoji 😂

```
map([🐮, 🍠, 🐔, 🌽], cook)
=> [🍔, 🍟, 🍗, 🍿]

filter([🍔, 🍟, 🍗, 🍿], isVegetarian)
=> [🍟, 🍿]

reduce([🍔, 🍟, 🍗, 🍿], eat)
=> 💩
```

Why Immutability?

- “Safer”
- Improves update performance
- Required by some libraries (ex. Redux)

Recap: Five buckets o’ React

Functions

Objects

Classes

Modules

Immutables

Takeaways

- JSX === React.createElement()
- Elements (and Element Trees) === objects
- Use classes for state, functions for everything else
- One component per .js, one .css per component
- Don’t mutate data

bit.ly/jazzy-que

Thanks!

- bit.ly/jazzy-que - example code and learning resource for React and React Storybook
- @radishmouse
- bignerdranch.com
- Bootcamps: August 14-18 and October 23-27
- Use "JAZZCON2017" for 10% off!