REASON

ReasonReact

Stateless Component

Greeting.re

```
let component = ReasonReact.statelessComponent("Greeting");

let make = (_children) => {
  ...component,
  render: _self => <h1>(ReasonReact.string("Hello"))</h1>,
};
```

App.re

```
ReactDOMRe.renderToElementWithId(<Greeting />, "root");
```

Props

Greeting.re

```
let component = ReasonReact.statelessComponent("Greeting");

let make = (~name, _children) => {
  ...component,
  render: _self => <h1>(ReasonReact.string(“Hello " ++ name))</h1>,
};
```

App.re

```
ReactDOMRe.renderToElementWithId(<Greeting name="Helsinki" />, "root");
```

Props

Greeting.re

```
let component [illegible]asonReact.statelessComponent("Greeting");

let make = (~name, _children) => {
  ...component,
  render: _self => <h1>(ReasonReact.string("Hello " ++ name))</h1>,
};
```

App.re

```
ReactDOMRe.renderToElementWithId(<Greeting name=“Odessa" />, "root");
```

Props

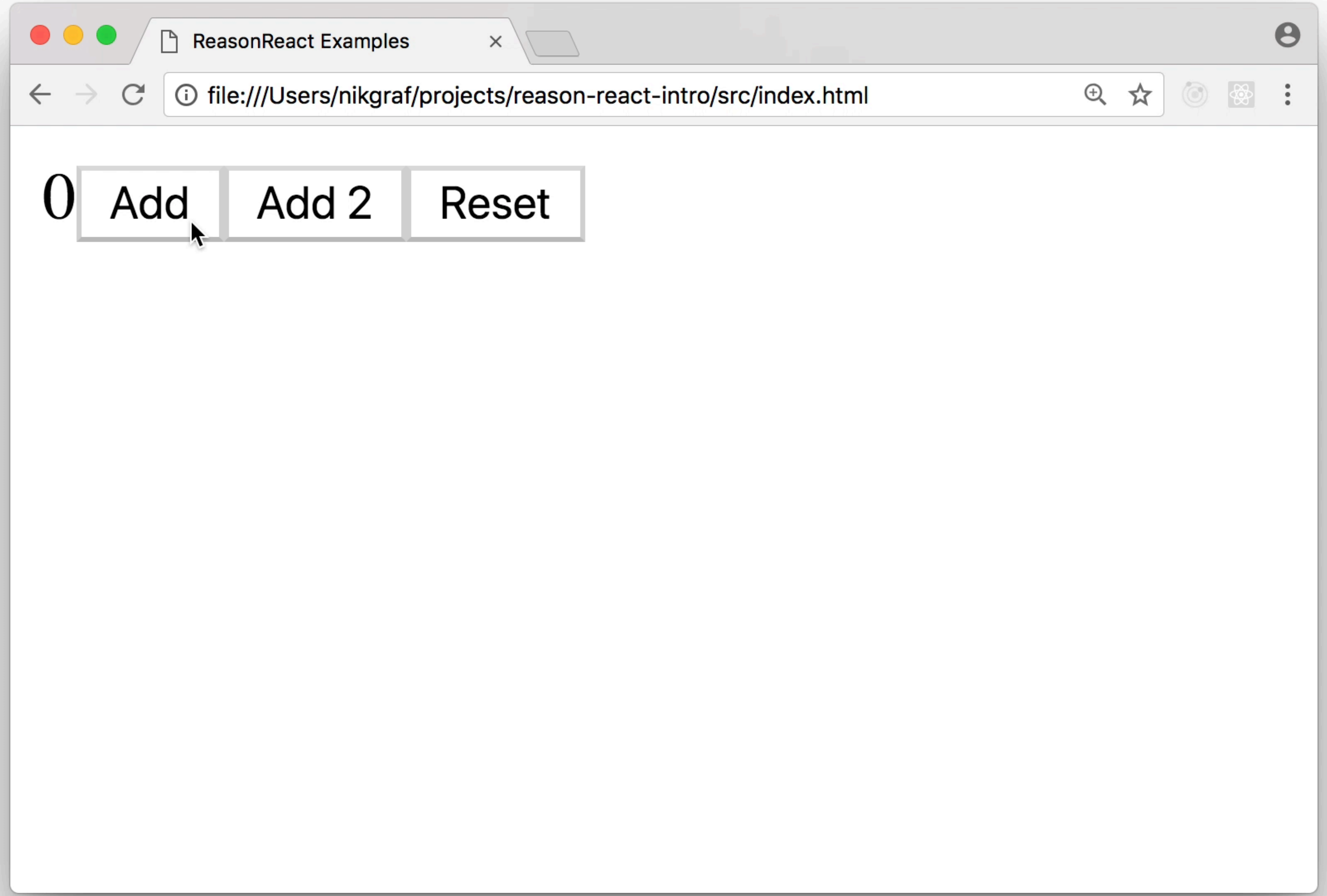
Greeting.re

```
let component = ReasonReact.statelessComponent("Greeting");

let make = (~name, _children) => {
  ...component,
  render: _self => <h1>(ReasonReact.string("Hello " ++ name))</h1>,
};
```

App.re

```
ReactDOMRe.renderToElementWithId(<Greeting name="Odessa" />, "root");
```


ReasonReact Examples
file:///Users/nikgraf/projects/reason-react-intro/src/index.html
0
Add
Add 2
Reset

```
type state = {count: int};
type state = {count: int};

type action =
  | Add(int)
  | Reset;
```

```reason
type state = {count: int};

type action =
  | Add(int)
  | Reset;

let s = ReasonReact.string;
```

```
type state = {count: int};

type action =
  | Add(int)
  | Reset;

let s = ReasonReact.string;

let component = ReasonReact.reducerComponent("Counter");
```

```
type state = {count: int};

type action =
  | Add(int)
  | Reset;

let s = ReasonReact.string;

let component = ReasonReact.reducerComponent("Counter");

let make = _children => {
  ...component,
  initialState: () => {count: 0},
type state = {count: int};

type action =
  | Add(int)
  | Reset;

let s = ReasonReact.string;

let component = ReasonReact.reducerComponent("Counter");

let make = _children => {
  ...component,
  initialState: () => {count: 0},
  reducer: (action, state) =>
    switch (action) {
    | Add(value) => ReasonReact.Update({count: state.count + value})
    | Reset => ReasonReact.Update({count: 0})
    },
type state = {count: int};

type action =
  | Add(int)
  | Reset;

let s = ReasonReact.string;

let component = ReasonReact.reducerComponent("Counter");

let make = _children => {
  ...component,
  initialState: () => {count: 0},
  reducer: (action, state) =>
    switch (action) {
    | Add(value) => ReasonReact.Update({count: state.count + value})
    | Reset => ReasonReact.Update({count: 0})
    },
  render: self =>
    <div>
      (s(string_of_int(self.state.count)))
      <button onClick=(_event => self.send(Add(1)))> (s(“Add")) </button>
```

```
let s = ReasonReact.string;

let component = ReasonReact.reducerComponent("Counter");

let make = _children => {
  ...component,
  initialState: () => {count: 0},
  reducer: (action, state) =>
    switch (action) {
    | Add(value) => ReasonReact.Update({count: state.count + value})
    | Reset => ReasonReact.Update({count: 0})
    },
  render: self =>
    <div>
      (s(string_of_int(self.state.count)))
      <button onClick=(_event => self.send(Add(1)))> (s(“Add")) </button>
      <button onClick=(_event => self.send(Add(2)))> (s("Add 2")) </button>
      <button onClick=(_event => self.send(Reset))> (s("Reset")) </button>
    </div>,
};
```

Interop with JavaScript

BuckleScript allows us to write bindings

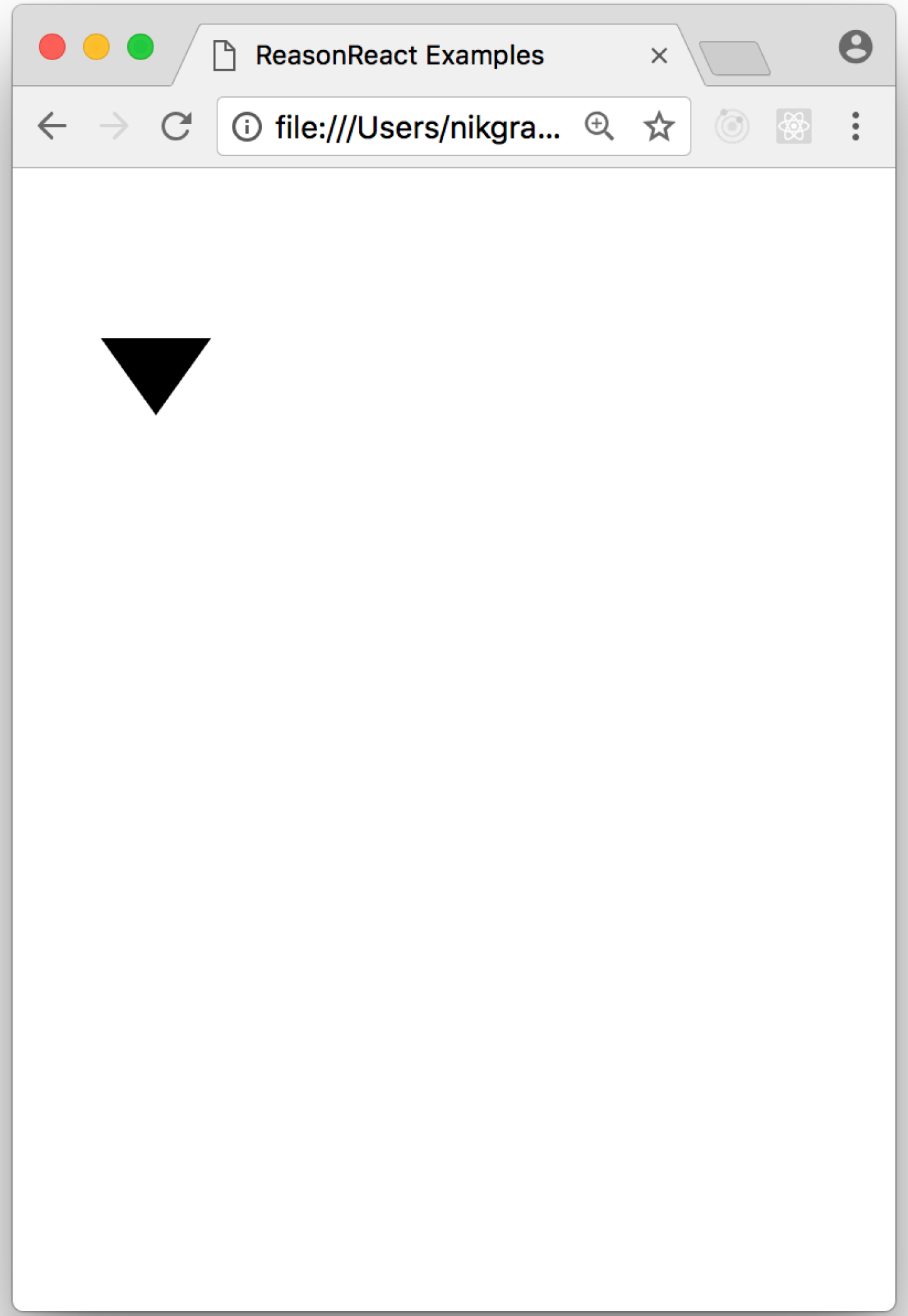
ReasonReact

- wrapJsForReason
- wrapReasonForJs

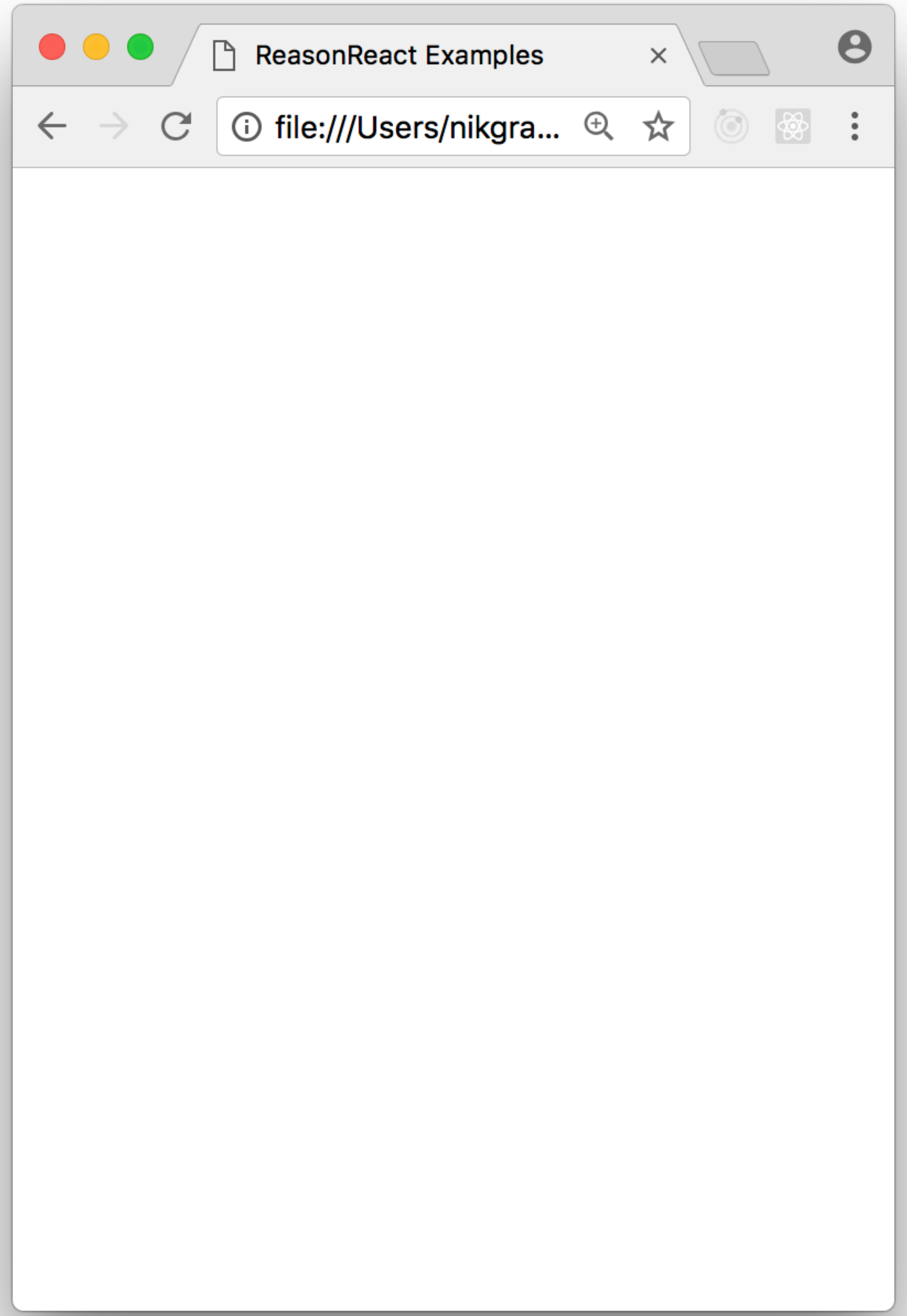
```
[@bs.module "rebass"] external jsArrow : ReasonReact.reactClass = "Arrow";

let make = (~direction: string, children) =>
  ReasonReact.wrapJsForReason(
    ~reactClass=jsArrow,
    ~props={"direction": direction},
    children,
  );
```

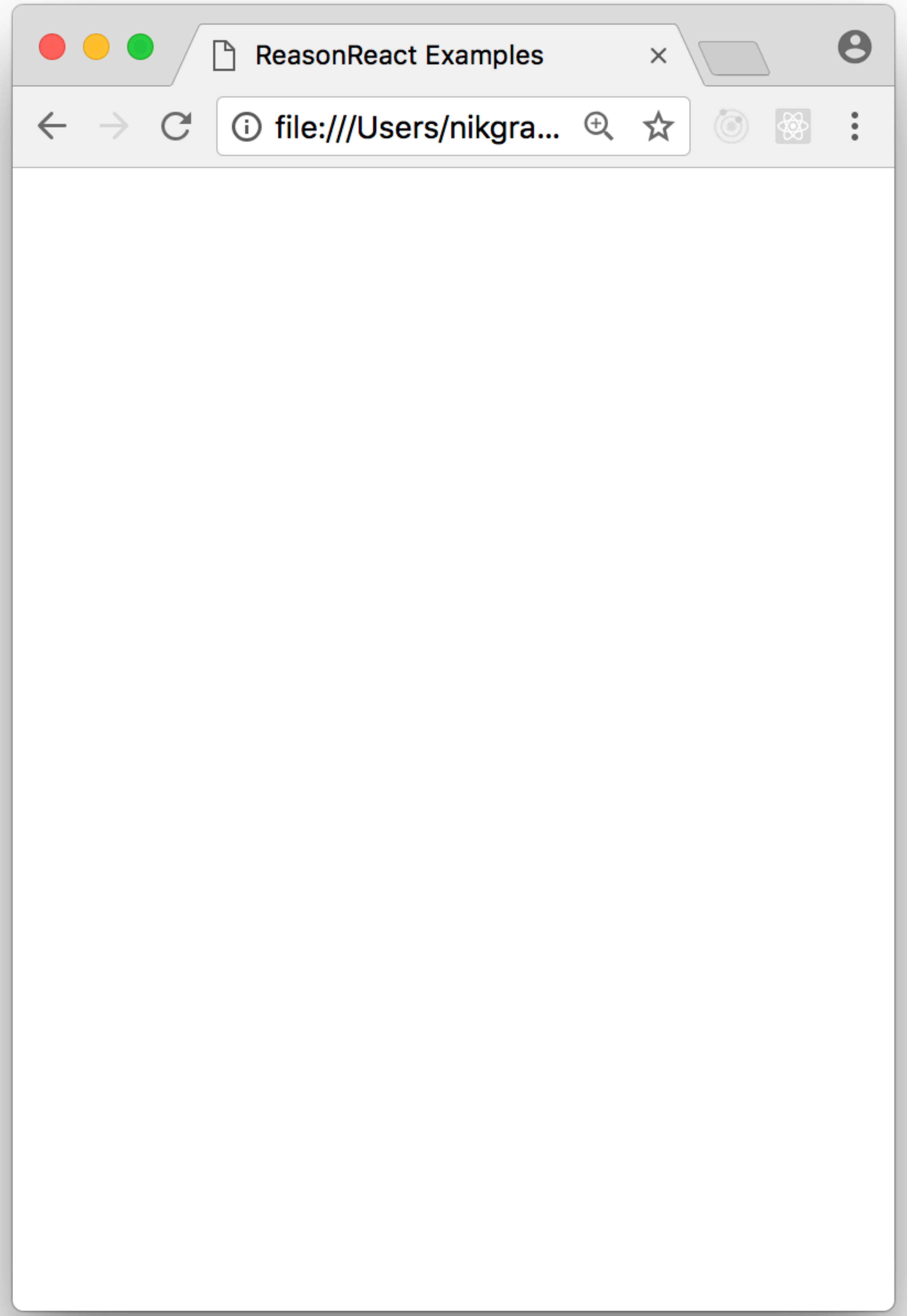
```
<Arrow direction="down" />
```

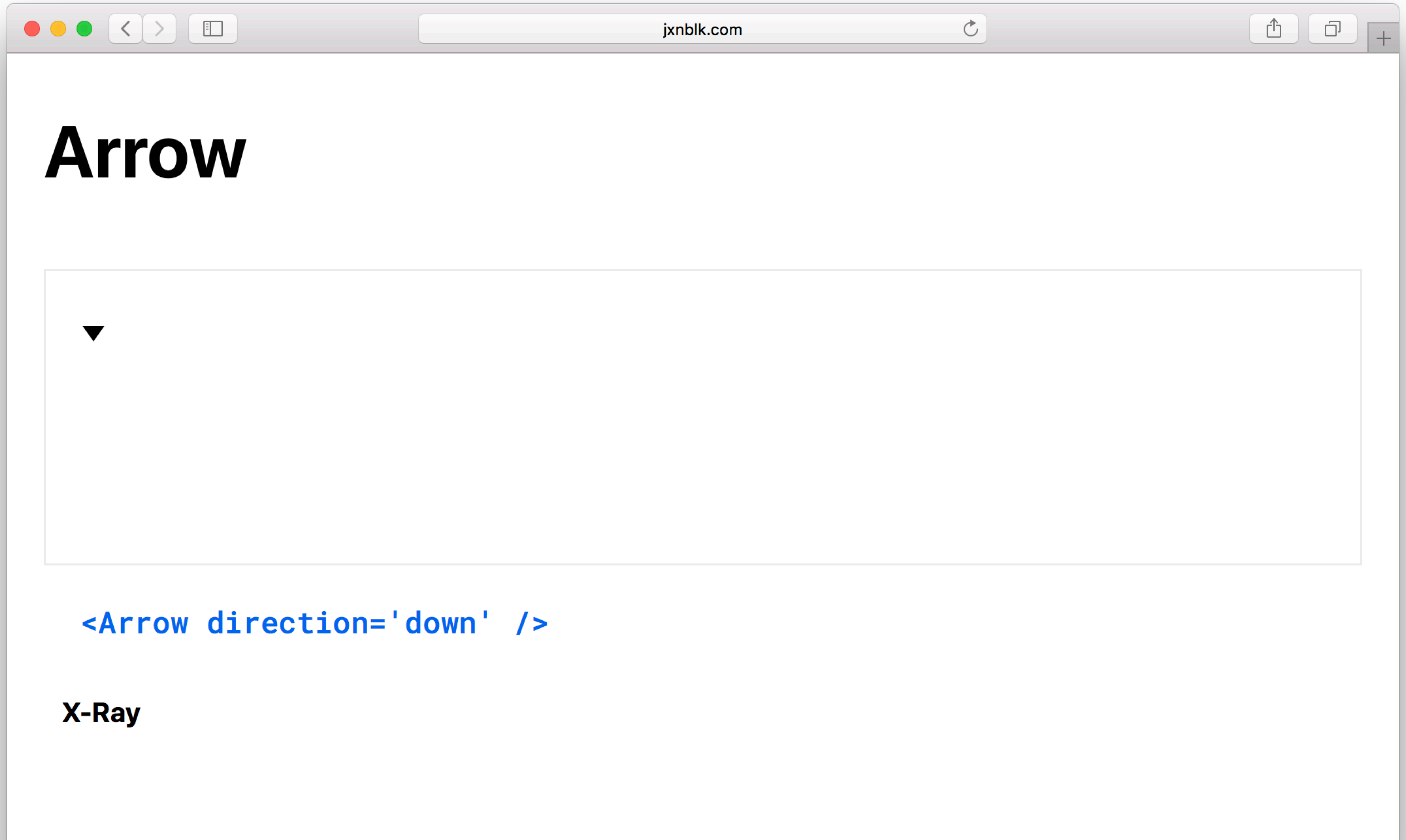

ReasonReact Examples
file:///Users/nikgra...

```
<Arrow direction="left" />
```


ReasonReact Examples
file:///Users/nikgra...

```
<Arrow direction="notRight"/>
```


ReasonReact Examples
file:///Users/nikgra...



Arrow

▼

```
<Arrow direction='down' />
```

X-Ray

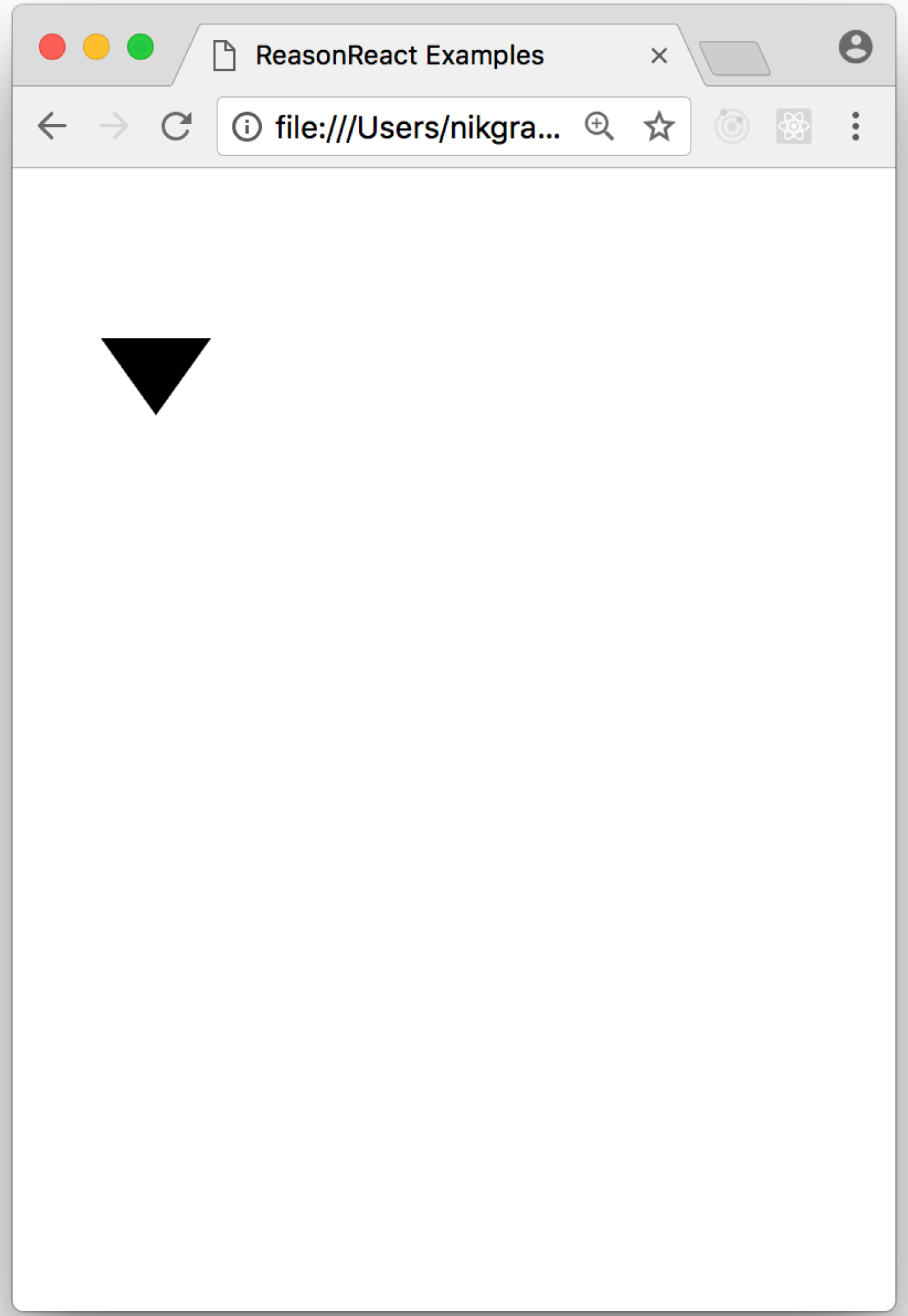
Variants to the rescue!

```
[@bs.module "rebass"] external jsArrow : ReasonReact.reactClass = "Arrow";

type direction =
  | Up
  | Down;

let make = (~direction, children) => {
  let directionString =
    switch (direction) {
    | Up => "up"
    | Down => "down"
    };
  ReasonReact.wrapJsForReason(
    ~reactClass=jsArrow,
    ~props={"direction": directionString},
    children,
  );
};
```

```
<Arrow direction=Arrow.Down />;
```


ReasonReact Examples
file:///Users/nikgra...

```
<Arrow direction=Arrow.Left />;
```

```
<Arrow direction=Arrow.Left />;
```

```
3 | let make = _children => {
4 |   ...component,
5 |   render: _self => <div> <Arrow direction=Arrow.Left /> </div>,
6 | };

The variant constructor Arrow.Left can't be found.
```

The End