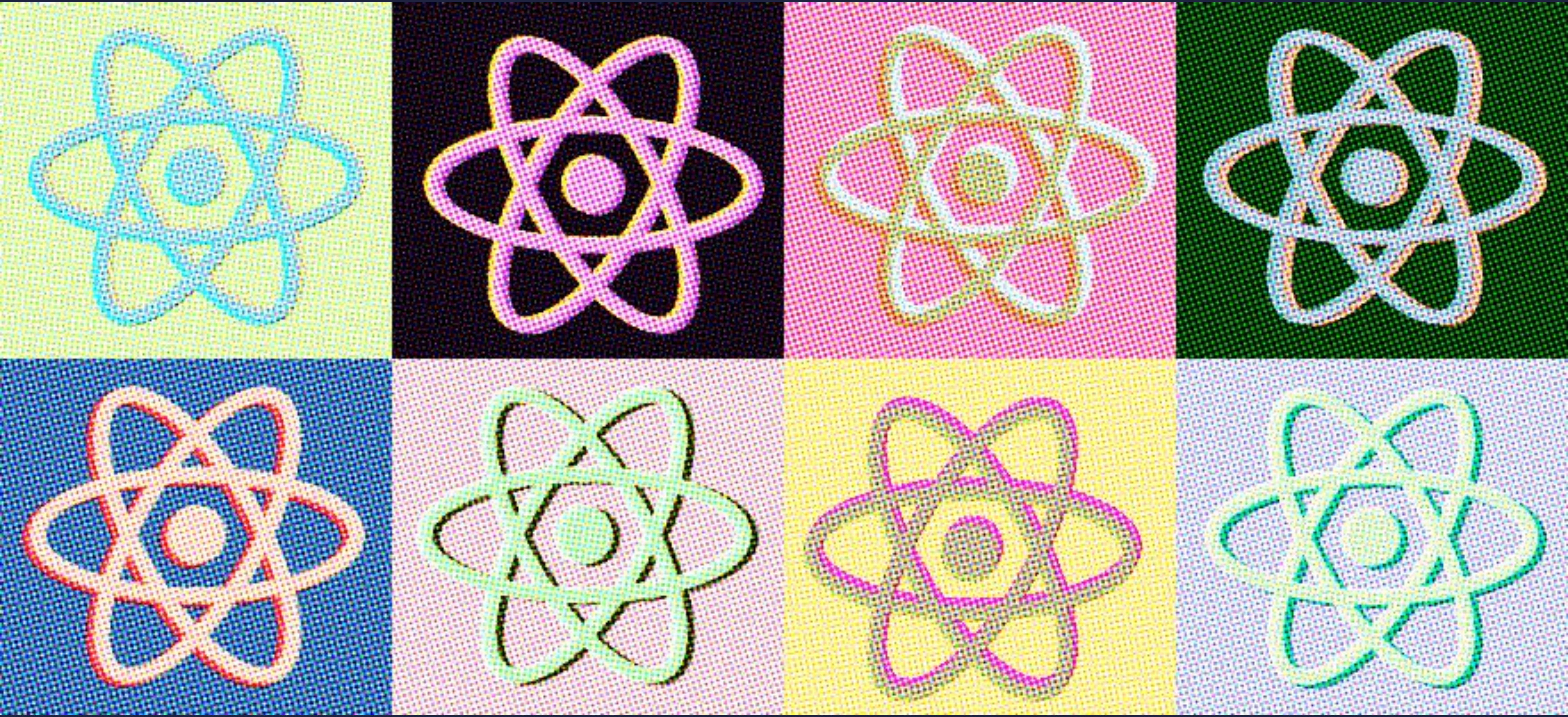


REACT-HASKELL

HI, MY NAME IS JOEL

HI, MY NAME IS JOEL

... AND I USE HASKELL

HASKELL

- statically typed
- lazy
- purely functional

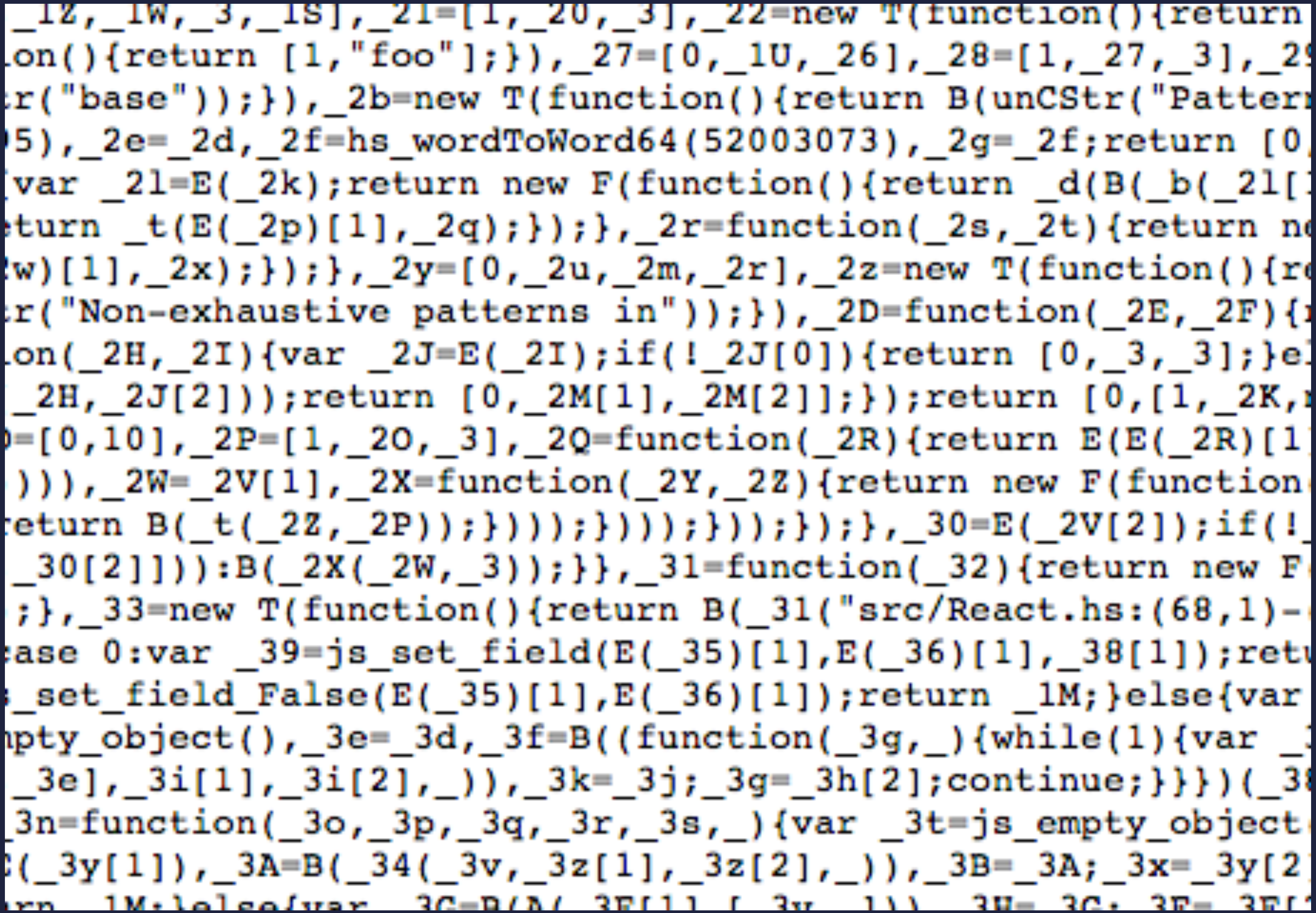
HASKELL

- ~~statically typed~~
- ~~lazy~~
- ~~purely functional~~
- really cool

HASTE

- dialect of Haskell
- runs in browser

HASTE



```
_1Z,_1W,_3,_1S],_21=[1,_20,_3],_22=new T(function(){return
on(){return [1,"foo"];}),_27=[0,_1U,_26],_28=[1,_27,_3],_2
r("base"));}),_2b=new T(function(){return B(unCStr("Patter
5),_2e=_2d,_2f=hs_wordToWord64(52003073),_2g=_2f;return [0
var _2l=E(_2k);return new F(function(){return _d(B(_b(_2l[
turn _t(E(_2p)[1],_2q);});},_2r=function(_2s,_2t){return n
w)[1],_2x);});},_2y=[0,_2u,_2m,_2r],_2z=new T(function(){r
r("Non-exhaustive patterns in"));}),_2D=function(_2E,_2F){
on(_2H,_2I){var _2J=E(_2I);if(!_2J[0]){return [0,_3,_3];}e
_2H,_2J[2]));return [0,_2M[1],_2M[2]];});return [0,[1,_2K,
=[0,10],_2P=[1,_2O,_3],_2Q=function(_2R){return E(E(_2R)[1
))),_2W=_2V[1],_2X=function(_2Y,_2Z){return new F(function
eturn B(_t(_2Z,_2P));})));})));}));});},_30=E(_2V[2]);if(!
_30[2]])):B(_2X(_2W,_3));}},_31=function(_32){return new F
;},_33=new T(function(){return B(_31("src/React.hs:(68,1)-
ase 0:var _39=js_set_field(E(_35)[1],E(_36)[1],_38[1]);ret
_set_field_False(E(_35)[1],E(_36)[1]);return _1M;}else{var
pty_object(),_3e=_3d,_3f=B((function(_3g,_){while(1){var _
_3e],_3i[1],_3i[2],_)),_3k=_3j;_3g=_3h[2];continue;}}})(_3
_3n=function(_3o,_3p,_3q,_3r,_3s,_){var _3t=js_empty_object
(_3y[1]),_3A=B(_34(_3v,_3z[1],_3z[2],_)),_3B=_3A;_3x=_3y[2
```

BLAZE-HTML

```
sample :: Html
sample = p ! class_ "styled" $ em "Basic Algebra"
```

becomes

```
<p class="styled">
    <em>Basic Algebra</em>
</p>
```

REACT-HASKELL

```haskell
sample :: React
sample = p <! className "styled" $ em "Basic Algebra"
```

becomes

```html
<p class="styled">
    <em>Basic Algebra</em>
</p>
```

PUT IT ON THE PAGE

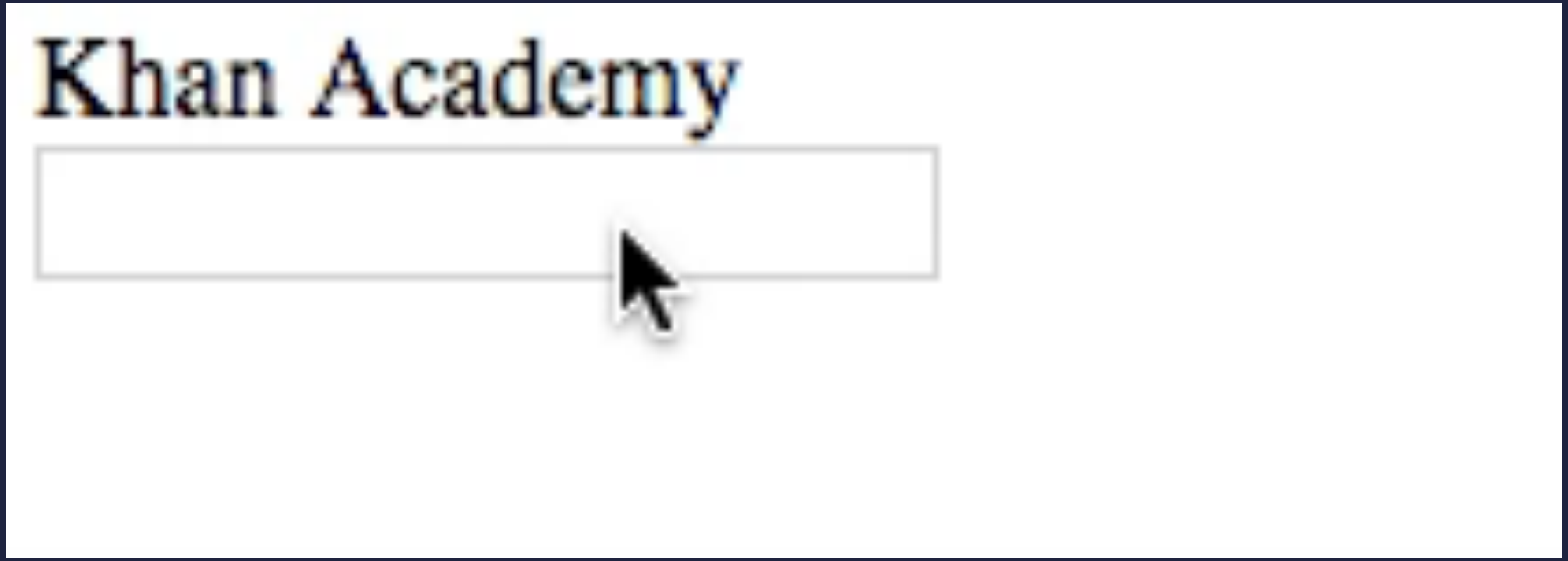
```haskell
main :: IO ()
main = do
    Just elem <- elemById "id"
    render elem sample
```

MORE COMPLICATED

```
sample :: React
sample = div <! className "beautify" $ do
    "Khan Academy"

    input

    "Rewritten in Haskell"
```

CONTROLLED COMPONENT

```
view :: Elem -> JSString -> IO ()
view elem str = render elem $
    div $ do
        "Khan Academy"

        input <! onChange (view elem . targetValue)

        text str
```


Khan Academy

LET'S MAKE THAT EASIER

STATEFUL COMPONENT REDUX

```
view :: StatefulReact JSString
view = div $ do
    "Khan Academy"

    input <! onChange' (updateState . targetValue)

    text str
```

LIFECYCLE METHODS

`componentDidMount`, `componentWillUnmount`, ...

?

QUESTIONS?