{codemotion}

React, The Inglorious Way

Matteo Antony Mistretta

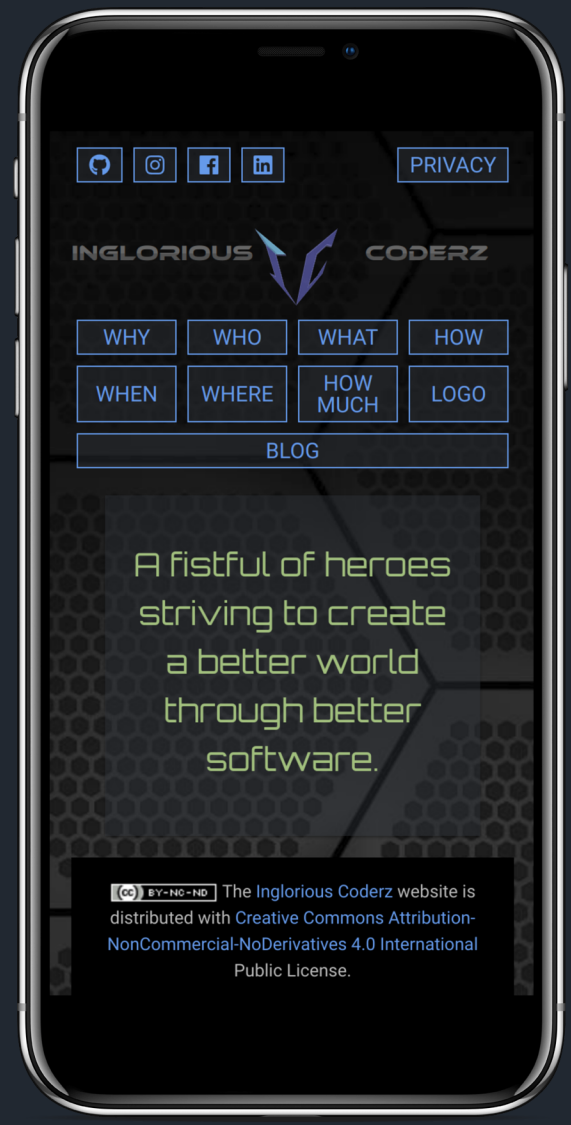
Inglorious Coderz

@antonymistretta

Why

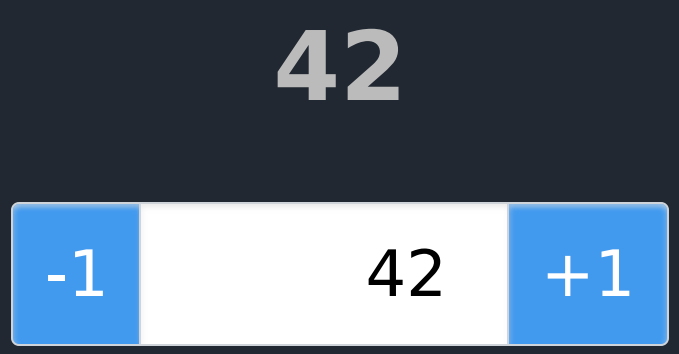
- React is evolving rapidly
- A few rules, lots of strategies
- Learning them makes us better coders

```
antony@ingloriouscoderz ~> whoami
```

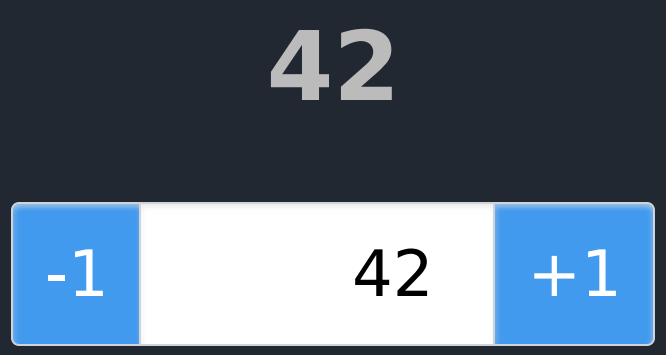


Agenda

- *Class Components*
- Container/Presentational
- Higher-Order Components
- Render Props
- Hooks



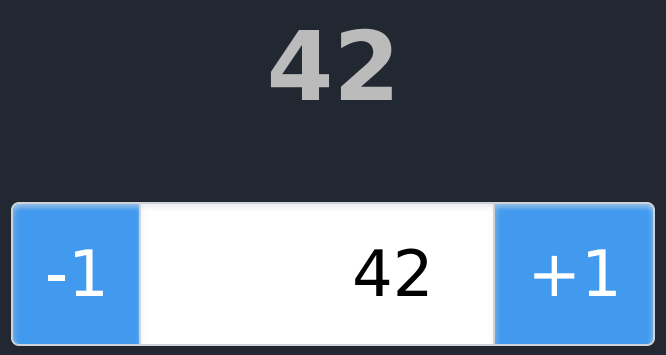

```
class Counter extends Component {
  constructor(props) {
    super(props)

    this.state = { count: props.initialCount }
    this.increment = this.increment.bind(this)
  }

  increment() {
    this.setState({ count: this.state.count + 1 })
  }

  decrement() {
    this.setState({ count: this.state.count - 1 })
  }

  render() {
    const { count } = this.state

    return (
      <div>
        <h1>{count}</h1>
        <div className="input-group">
          <button onClick={this.decrement.bind(this)}>-1</button>
          <input
            type="number"
            value={count}
            onChange={event => {
              this.setState({ count: parseInt(event.target.value) })
            }}
          />
          <button onClick={this.increment}>+1</button>
        </div>
      </div>
    )
  }
}

render(<Counter initialCount={42} />)
```



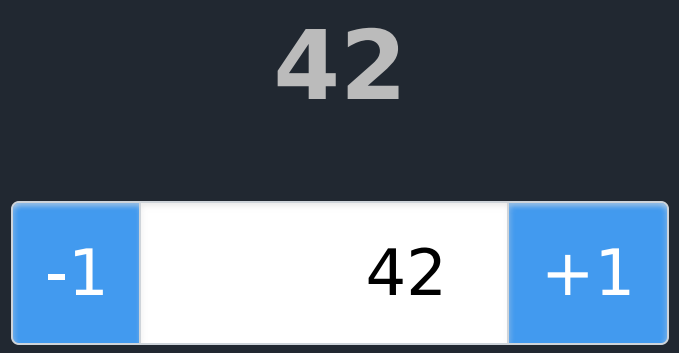

```jsx
class Counter extends PureComponent {
  state = { count: this.props.initialCount }

  increment = () => this.setState(({ count }) => ({ count: count + 1 }))
  decrement = () => this.setState(({ count }) => ({ count: count - 1 }))
  setCount = count => this.setState({ count })

  handleChange = event => this.setCount(parseInt(event.target.value))

  render() {
    const { count } = this.state

    return (
      <>
        <h1>{count}</h1>
        <div className="input-group">
          <button onClick={this.decrement}>-1</button>
          <input type="number" value={count} onChange={this.handleChange} />
          <button onClick={this.increment}>+1</button>
        </div>
      </>
    )
  }
}

render(<Counter initialCount={42} />)
```

Agenda

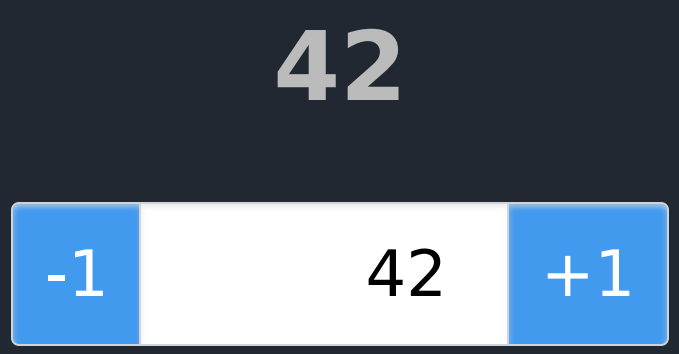
- Class Components
- *Container/Presentational*
- Higher-Order Components
- Render Props
- Hooks




```jsx
class Counter extends PureComponent {
  state = { count: this.props.initialCount }

  increment = () => this.setState(({ count }) => ({ count: count + 1 }))
  decrement = () => this.setState(({ count }) => ({ count: count - 1 }))
  setCount = count => this.setState({ count })

  handleChange = event => this.setCount(parseInt(event.target.value))

  render() {
    const { count } = this.state

    return (
      <>
        <h1>{count}</h1>
        <div className="input-group">
          <button onClick={this.decrement}>-1</button>
          <input type="number" value={count} onChange={this.handleChange} />
          <button onClick={this.increment}>+1</button>
        </div>
      </>
    )
  }
}

render(<Counter initialCount={42} />)
```




```jsx
class CounterContainer extends PureComponent {
  state = { count: this.props.initialCount }

  increment = () => this.setState(({ count }) => ({ count: count + 1 }))
  decrement = () => this.setState(({ count }) => ({ count: count - 1 }))
  setCount = count => this.setState({ count })

  handleChange = event => this.setCount(parseInt(event.target.value))

  render() {
    return (
      <Counter
        count={this.state.count}
        increment={this.increment}
        decrement={this.decrement}
        handleChange={this.handleChange}
      />
    )
  }
}

function Counter({ count, increment, decrement, handleChange }) {
  return (
    <>
      <h1>{count}</h1>
      <div className="input-group">
        <button onClick={decrement}>-1</button>
        <input type="number" value={count} onChange={handleChange} />
        <button onClick={increment}>+1</button>
      </div>
    </>
  )
}

render(<CounterContainer initialCount={42} />)
```




```jsx
class CounterContainer extends PureComponent {
  state = { count: this.props.initialCount }

  increment = () => this.setState(({ count }) => ({ count: count + 1 }))
  decrement = () => this.setState(({ count }) => ({ count: count - 1 }))
  setCount = count => this.setState({ count })

  handleChange = event => this.setCount(parseInt(event.target.value))

  render() {
    return Counter({
      count: this.state.count,
      increment: this.increment,
      decrement: this.decrement,
      handleChange: this.handleChange,
    })
  }
}

function Counter({ count, increment, decrement, handleChange }) {
  return (
    <>
      <h1>{count}</h1>
      <div className="input-group">
        <button onClick={decrement}>-1</button>
        <input type="number" value={count} onChange={handleChange} />
        <button onClick={increment}>+1</button>
      </div>
    </>
  )
}

render(<CounterContainer initialCount={42} />)
```

Agenda

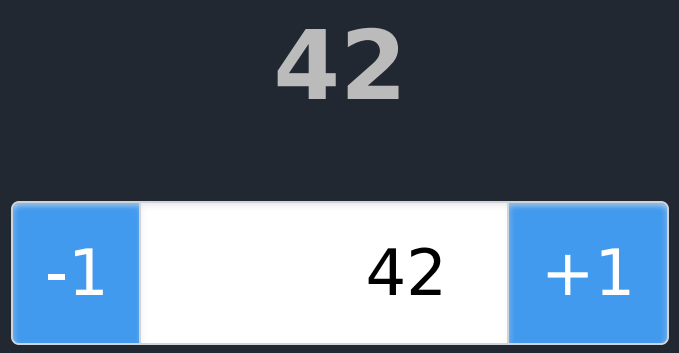
- Class Components
- Container/Presentational
- *Higher-Order Components*
- Render Props
- Hooks

Hello world!

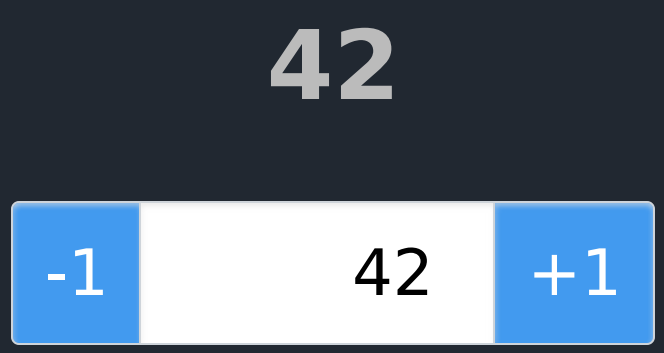
```
function Parent() {
  return <Child />
}

function Child() {
  return 'Hello world!'
}

const enhance = Enhanced => {
  return function Wrapper(props) {
    return (
      <h1>
        <Enhanced {...props} />
      </h1>
    )
  }
}

Child = enhance(Child)

render(Parent)
```

Hello WORLD!

```
function Parent() {
  return <Child />
}

function Child({ who }) {
  return `Hello ${who}!`
}

const enhance = who => Enhanced => {
  return function Wrapper(props) {
    const shoutedWho = who.toUpperCase()
    return (
      <h1>
        <Enhanced {...props} who={shoutedWho} />
      </h1>
    )
  }
}

Child = enhance('world')(Child)

render(Parent)
```




```jsx
class CounterContainer extends PureComponent {
  state = { count: this.props.initialCount }

  increment = () => this.setState(({ count }) => ({ count: count + 1 }))
  decrement = () => this.setState(({ count }) => ({ count: count - 1 }))
  setCount = count => this.setState({ count })

  handleChange = event => this.setCount(parseInt(event.target.value))

  render() {
    return (
      <Counter
        count={this.state.count}
        increment={this.increment}
        decrement={this.decrement}
        handleChange={this.handleChange}
      />
    )
  }
}

function Counter({ count, increment, decrement, handleChange }) {
  return (
    <>
      <h1>{count}</h1>
      <div className="input-group">
        <button onClick={decrement}>-1</button>
        <input type="number" value={count} onChange={handleChange} />
        <button onClick={increment}>+1</button>
      </div>
    </>
  )
}

render(<CounterContainer initialCount={42} />)
```




```
const enhance = compose(
  withState('count', 'setCount', ({ initialCount }) => initialCount),
  withHandlers({
    increment: ({ setCount }) => () => setCount(count => count + 1),
    decrement: ({ setCount }) => () => setCount(count => count - 1),
    handleChange: ({ setCount }) => event =>
      setCount(parseInt(event.target.value)),
  }),
  memo,
)

const Counter = enhance(({ count, increment, decrement, handleChange }) => (
  <>
    <h1>{count}</h1>
    <div className="input-group">
      <button onClick={decrement}>-1</button>
      <input type="number" value={count} onChange={handleChange} />
      <button onClick={increment}>+1</button>
    </div>
  </>
))

render(<Counter initialCount={42} />)
```

Agenda

- Class Components
- Container/Presentational
- Higher-Order Components
- *Render Props*
- Hooks

Hello world!

```
function Parent() {
  return (
    <Wrapper>
      <Child />
    </Wrapper>
  )
}

function Child() {
  return 'Hello world!'
}

function Wrapper({ children }) {
  return <h1>{children}</h1>
}

render(Parent)
```

Hello WORLD!

```
function Parent() {
  return <Wrapper Component={Child} who="world" />
}

function Child({ who }) {
  return `Hello ${who}!`
}

function Wrapper({ Component, who }) {
  const shoutedWho = who.toUpperCase()
  return (
    <h1>
      <Component who={shoutedWho} />
    </h1>
  )
}

render(Parent)
```

Hello WORLDz!

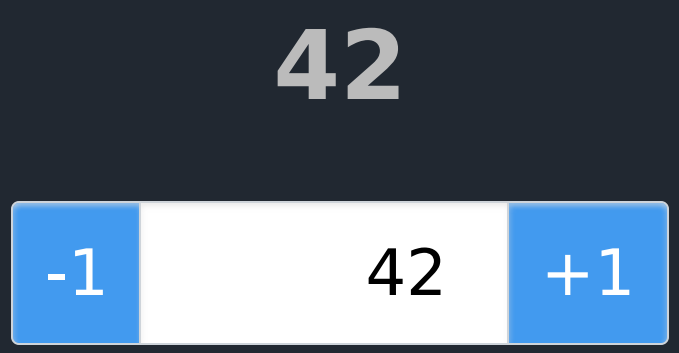
```
function Parent() {
  return <Wrapper render={who => <Child who={who + 'z'} />} who="world" />
}

function Child({ who }) {
  return `Hello ${who}!`
}

function Wrapper({ render, who }) {
  const shoutedWho = who.toUpperCase()
  return <h1>{render(shoutedWho)}</h1>
}

render(Parent)
```

Hello WORLDz!

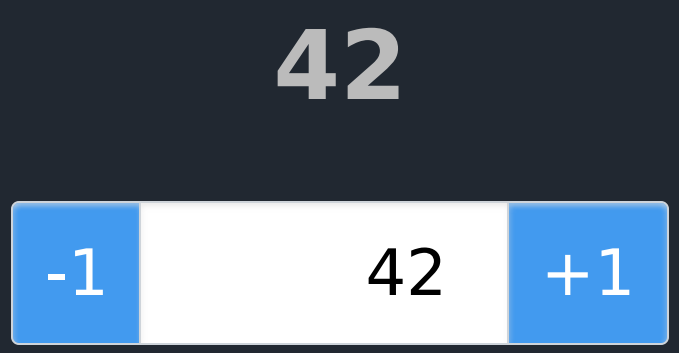
```
function Parent() {
  return <Wrapper who="world">{who => <Child who={who + 'z'} />}</Wrapper>
}

function Child({ who }) {
  return `Hello ${who}!`
}

function Wrapper({ children, who }) {
  const shoutedWho = who.toUpperCase()
  return <h1>{children(shoutedWho)}</h1>
}

render(Parent)
```

- world
- though
- ly
- tual
- issey

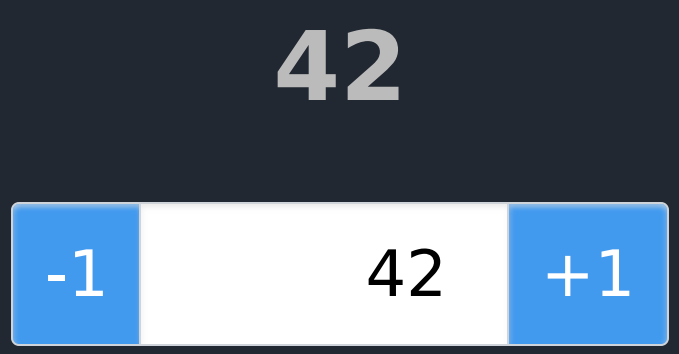
```
const Parent = ({ whos, simple }) =>
  simple ? (
    <SimpleList whos={whos} />
  ) : (
    <Wrapper
      renderOdd={who => <li style={styles.odd}>{'Odd ' + who}</li>}
      renderEven={who => <li style={styles.even}>{'Even ' + who}</li>}
      whos={whos}
    />
  )

const SimpleList = ({ whos }) => (
  <ul>
    {whos.map(who => (
      <li>{who}</li>
    ))}
  </ul>
)

const Wrapper = ({ renderOdd, renderEven, whos }) => (
  <ul>
    {whos.map((who, index) => (index % 2 ? renderEven(who) : renderOdd(who)))}
  </ul>
)

render(
  <Parent whos={['world', 'though', 'ly', 'tual', 'issey']} simple={true} />,
)

const styles = {
  odd: { color: 'grey' },
  even: { color: 'cornflowerblue' },
}
```




```jsx
class CounterContainer extends PureComponent {
  state = { count: this.props.initialCount }

  increment = () => this.setState(({ count }) => ({ count: count + 1 }))
  decrement = () => this.setState(({ count }) => ({ count: count - 1 }))
  setCount = count => this.setState({ count })

  handleChange = event => this.setCount(parseInt(event.target.value))

  render() {
    return (
      <Counter
        count={this.state.count}
        increment={this.increment}
        decrement={this.decrement}
        handleChange={this.handleChange}
      />
    )
  }
}

function Counter({ count, increment, decrement, handleChange }) {
  return (
    <>
      <h1>{count}</h1>
      <div className="input-group">
        <button onClick={decrement}>-1</button>
        <input type="number" value={count} onChange={handleChange} />
        <button onClick={increment}>+1</button>
      </div>
    </>
  )
}

render(<CounterContainer initialCount={42} />)
```



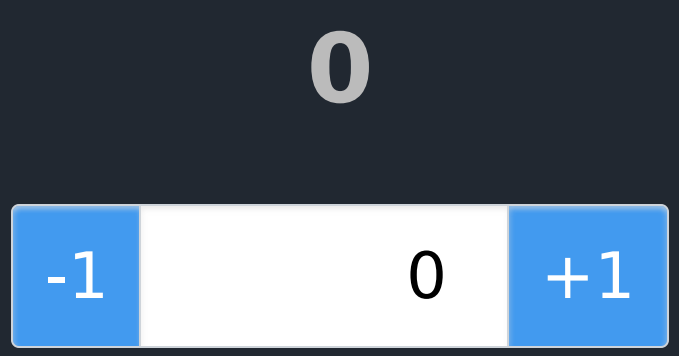

```jsx
class CounterContainer extends PureComponent {
  increment = () => this.setState(({ count }) => ({ count: count + 1 }))
  decrement = () => this.setState(({ count }) => ({ count: count - 1 }))
  setCount = count => this.setState({ count })

  handleChange = event => this.setCount(parseInt(event.target.value))

  state = {
    count: this.props.initialCount,
    increment: this.increment,
    decrement: this.decrement,
    handleChange: this.handleChange,
  }

  render() {
    return this.props.children(this.state)
  }
}

function Counter({ initialCount }) {
  return (
    <CounterContainer initialCount={initialCount}>
      {({ count, increment, decrement, handleChange }) => (
        <>
          <h1>{count}</h1>
          <div className="input-group">
            <button onClick={decrement}>-1</button>
            <input type="number" value={count} onChange={handleChange} />
            <button onClick={increment}>+1</button>
          </div>
        </>
      )}
    </CounterContainer>
  )
}

render(<Counter initialCount={42} />)
```



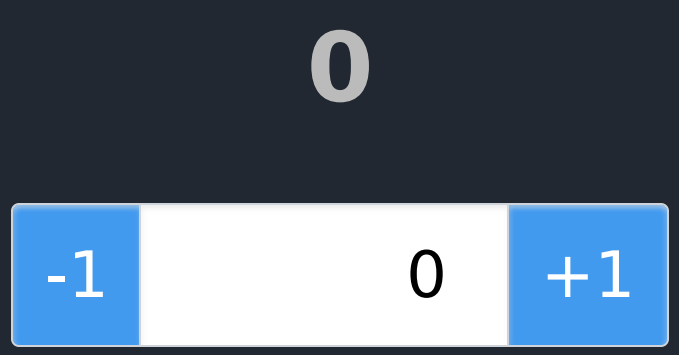

```jsx
class CounterContainer extends PureComponent {
  increment = () => this.setState(({ count }) => ({ count: count + 1 }))
  decrement = () => this.setState(({ count }) => ({ count: count - 1 }))
  setCount = count => this.setState({ count })

  handleChange = event => this.setCount(parseInt(event.target.value))

  state = {
    count: this.props.initialCount,
    increment: this.increment,
    decrement: this.decrement,
    handleChange: this.handleChange,
  }

  render() {
    return this.props.children(this.state)
  }
}

class Counter extends PureComponent {
  renderCounter = ({ count, increment, decrement, handleChange }) => (
    <>
      <h1>{count}</h1>
      <div className="input-group">
        <button onClick={decrement}>-1</button>
        <input type="number" value={count} onChange={handleChange} />
        <button onClick={increment}>+1</button>
      </div>
    </>
  )

  render() {
    return (
      <CounterContainer initialCount={this.props.initialCount}>
        {this.renderCounter}
      </CounterContainer>
    )
  }
}

render(<Counter initialCount={42} />)
```

Agenda

- Class Components
- Container/Presentational
- Higher-Order Components
- Render Props
- *Hooks*




```jsx
class CounterContainer extends PureComponent {
  increment = () => this.setState(({ count }) => ({ count: count + 1 }))
  decrement = () => this.setState(({ count }) => ({ count: count - 1 }))
  setCount = count => this.setState({ count })

  handleChange = event => this.setCount(parseInt(event.target.value))

  state = {
    count: 0,
    increment: this.increment,
    decrement: this.decrement,
    handleChange: this.handleChange,
  }

  render() {
    return this.props.children(this.state)
  }
}

function Counter() {
  return (
    <CounterContainer>
      {({ count, increment, decrement, handleChange }) => (
        <>
          <h1>{count}</h1>
          <div className="input-group">
            <button onClick={decrement}>-1</button>
            <input type="number" value={count} onChange={handleChange} />
            <button onClick={increment}>+1</button>
          </div>
        </>
      )}
    </CounterContainer>
  )
}

render(Counter)
```




```
function useCounter() {
  const [count, setCount] = useState(0)

  const increment = () => setCount(count + 1)
  const decrement = () => setCount(count - 1)

  const handleChange = event => setCount(parseInt(event.target.value))

  return { count, increment, decrement, handleChange }
}

function Counter() {
  const { count, increment, decrement, handleChange } = useCounter()

  return (
    <>
      <h1>{count}</h1>
      <div className="input-group">
        <button onClick={decrement}>-1</button>
        <input type="number" value={count} onChange={handleChange} />
        <button onClick={increment}>+1</button>
      </div>
    </>
  )
}

render(memo(Counter))
```

Which to use?

- HOCs: *before* render (change props, prevent renders)
- Render Props: mix logic / multiple sub-items / switch behavior
- Hooks: everything else

Thank you.

Questions?

html | pdf | source