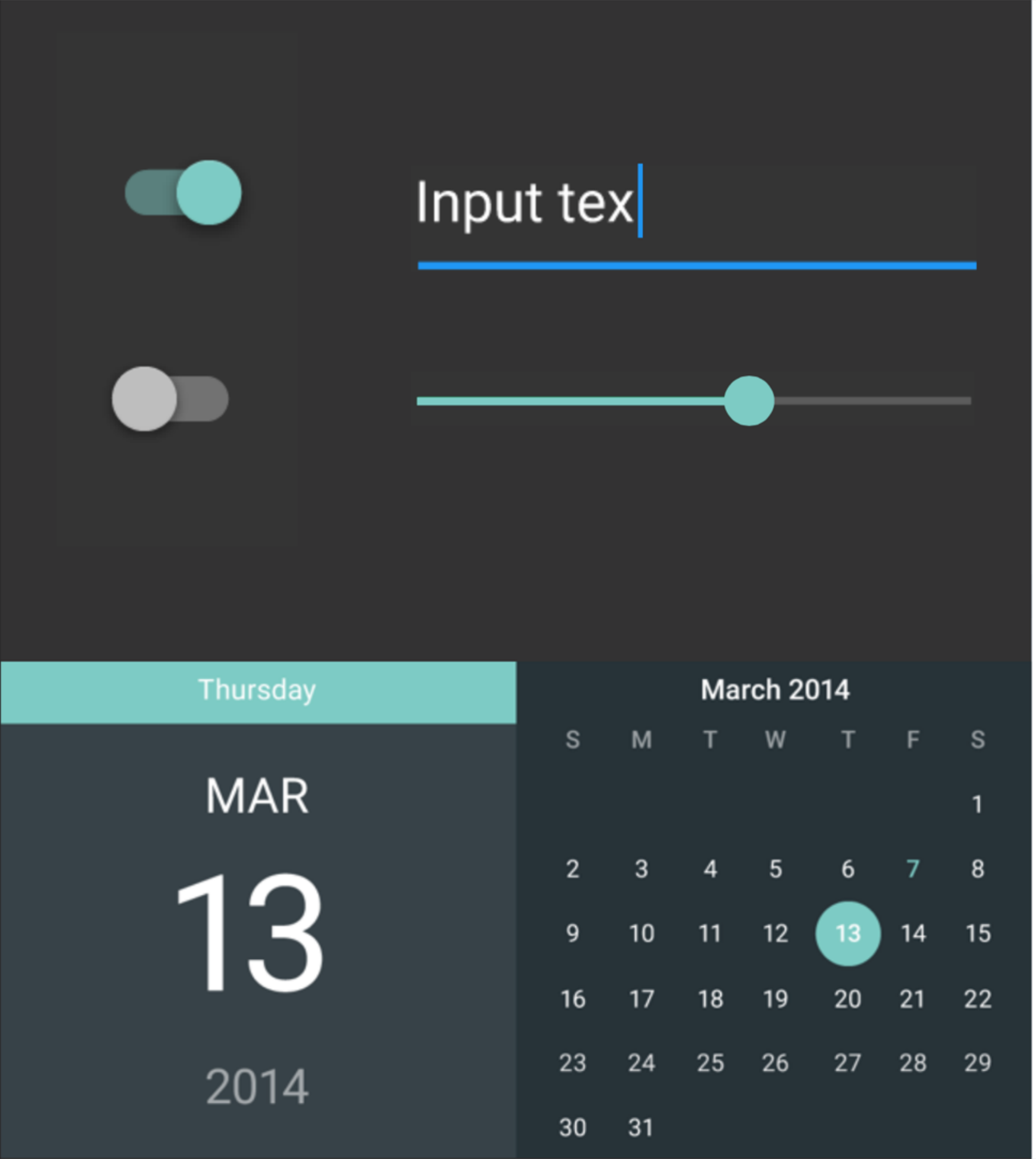
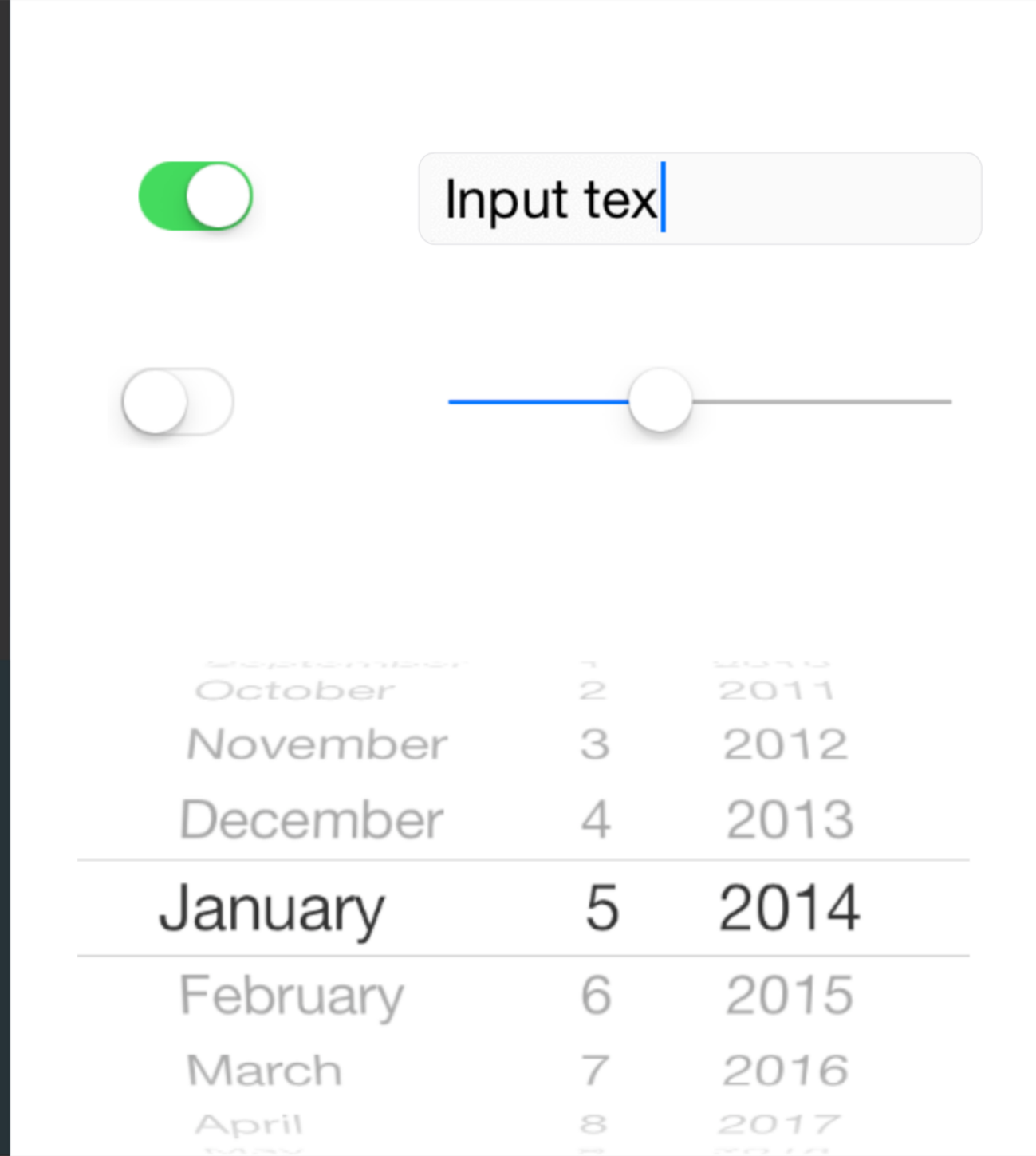
Why do we

native apps?

Native Apps

- Fast, responsive
- Complex gestures and smooth animations
- Consistent with platform


Input tex
Thursday
MAR
13
2014
March 2014
S M T W T F S
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31


Input tex
October 2 2011
November 3 2012
December 4 2013
January 5 2014
February 6 2015
March 7 2016
April 8 2017

Building native apps is hard

- Different stacks of technologies
- No knowledge and code sharing
- Slow iteration speed
- Hard to scale

Web got this right

Web

- ~~Different stacks of technologies~~ HTML / CSS / JS
- ~~No knowledge and code sharing~~ Same code and tech
- ~~Slow iteration speed~~ F5 / ⌘R
- ~~Hard to scale~~ React!

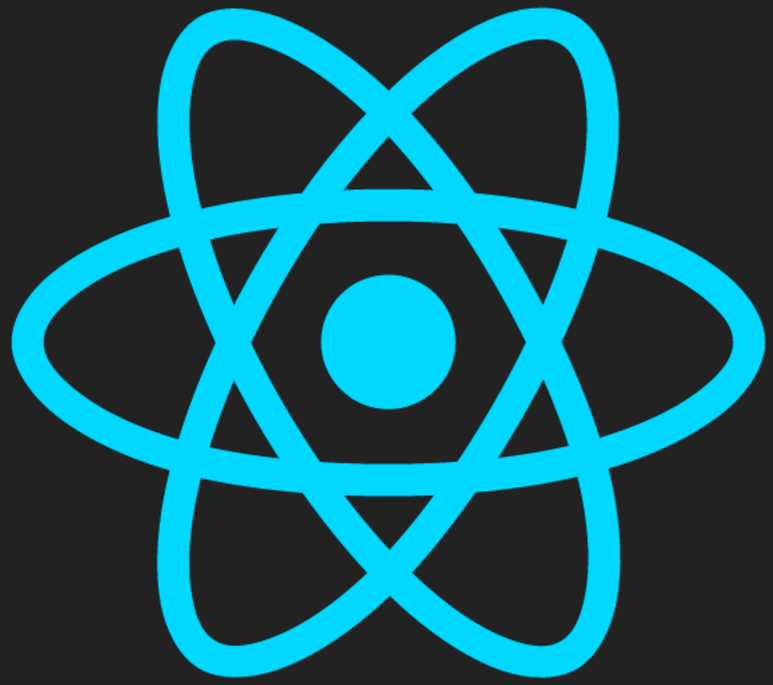
Web apps on the phone are not great

- Very hard to provide smooth experiences
- Not designed for complex interactions
- Impossible to embed native components

Development experience

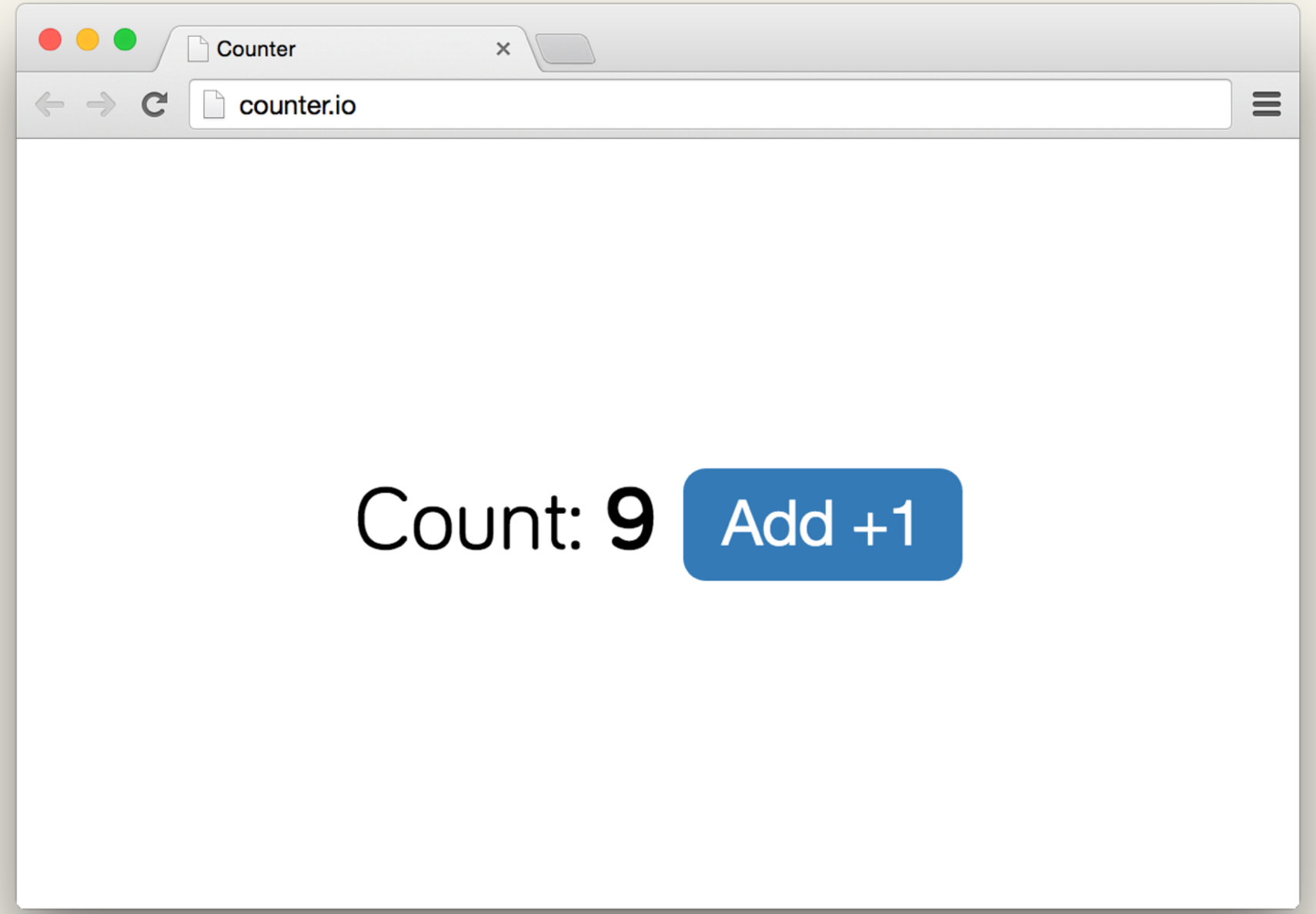
Awesome apps

React Native



$$UI = f^*(data)$$

* No side effects


Counter
counter.io
Count: 9
Add +1

$$UI = f(count)$$

```
UI = f(count) =
  div(
    span('Count ' + count),
    button('Add +1')
  )
```

```
render() {
  return (
    div(
      span(
        'Count: ' + b(this.state.count)
      ),
      button(
        'Add +1'
      )
    )
  )
}
```

```
render() {
  return (
    <div>
      <span>
        Count: <b>{this.state.count}</b>
      </span>
      <button>
        Add +1
      </button>
    </div>
  )
}
```

VirtualDOM

~~HTML~~

```
render() {
  return (
    <div>
      <span>
        Count: <b>{this.state.count}</b>
      </span>
      <button onClick={() => ??? }>
        Add +1
      </button>
    </div>
  )
}
```

Android

```
TextView text = (TextView)findViewByID(R.layout.label);
text.setText('10');
```

Objective-C

```
_label.text = @"10";
```

JavaScript

```
document.getElementByID('count').children[1].innerHTML = '10';
$('#counter b').html('10');
```

too complex

```
render() {
  var count = this.state.count;
  return (
    <div>
      <span>
        Count: <b>{count}</b>
      </span>
      <button onClick={() => ??? }>
        Add +1
      </button>
    </div>
  )
}
```

```
render() {
  var count = this.state.count;
  return (
    <div>
      <span>
        Count: <b>{count}</b>
      </span>
      <button onClick={() => this.setState({count: count + 1})}>
        Add +1
      </button>
    </div>
  )
}
```

setState

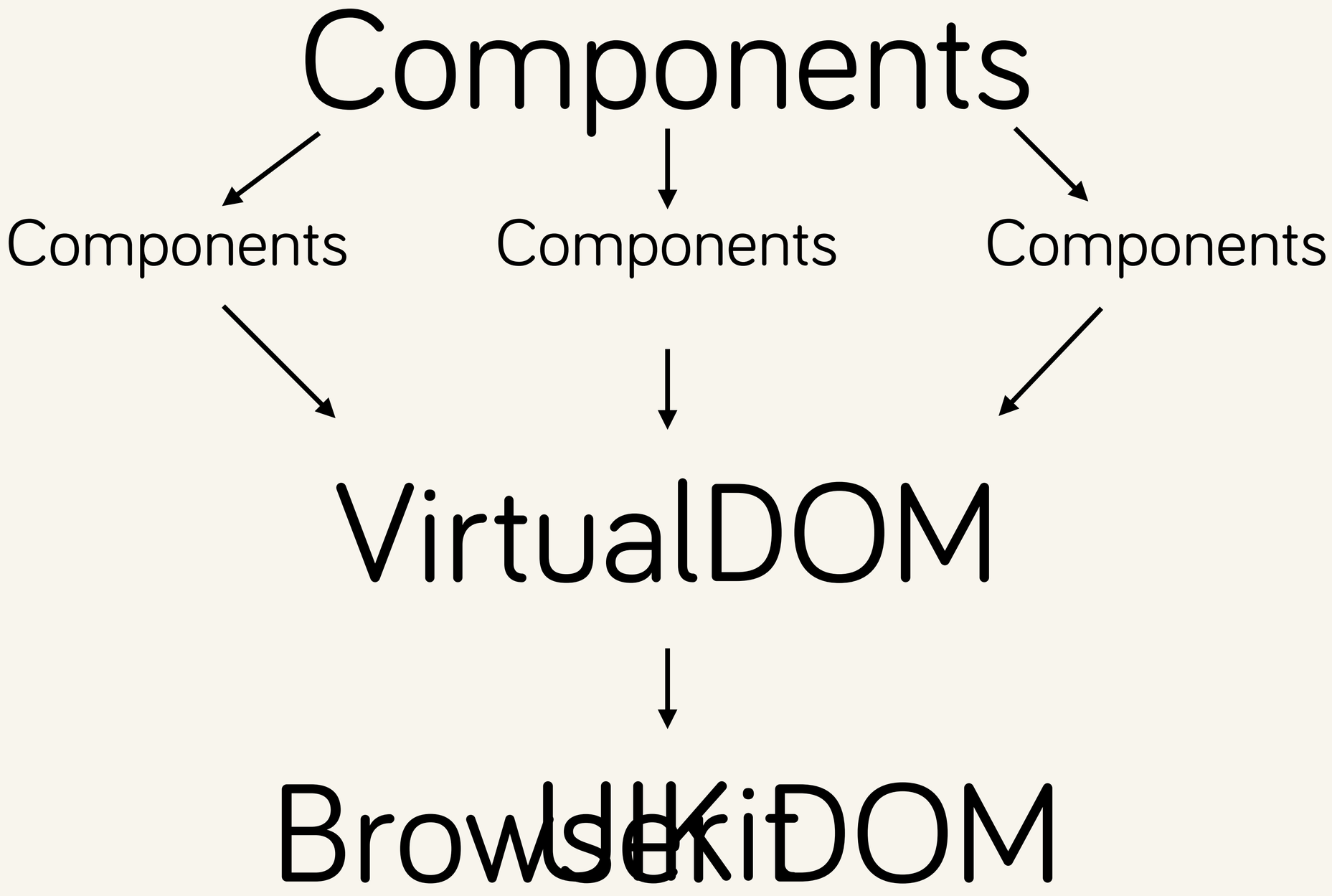
```
state = {count: 9}
```

```
<div>
   <span>
     Count: <b>9</b>
   </span>
   <button>
     Add +1
   </button>
 </div>
```

```
state = {count: 10}
```

```
<div>
   <span>
     Count: <b>10</b>
   </span>
   <button>
     Add +1
   </button>
 </div>
```

```
findDOMNode(b).innerHTML = '10';
```


Components
Components
Components
Components
VirtualDOM
BrowserKit
UIKit
DOM

JavaScript Core

- Part of WebKit project
- Open Source
- Ships with iOS

Runtime

`<div>`	`<View>`
`<span>`	`<Text>`
`<img>`	`<Image>`
	`<ScrollView>`
	`<MapView>`
	`<TabBar>`
	`<DatePicker>`
	`...`

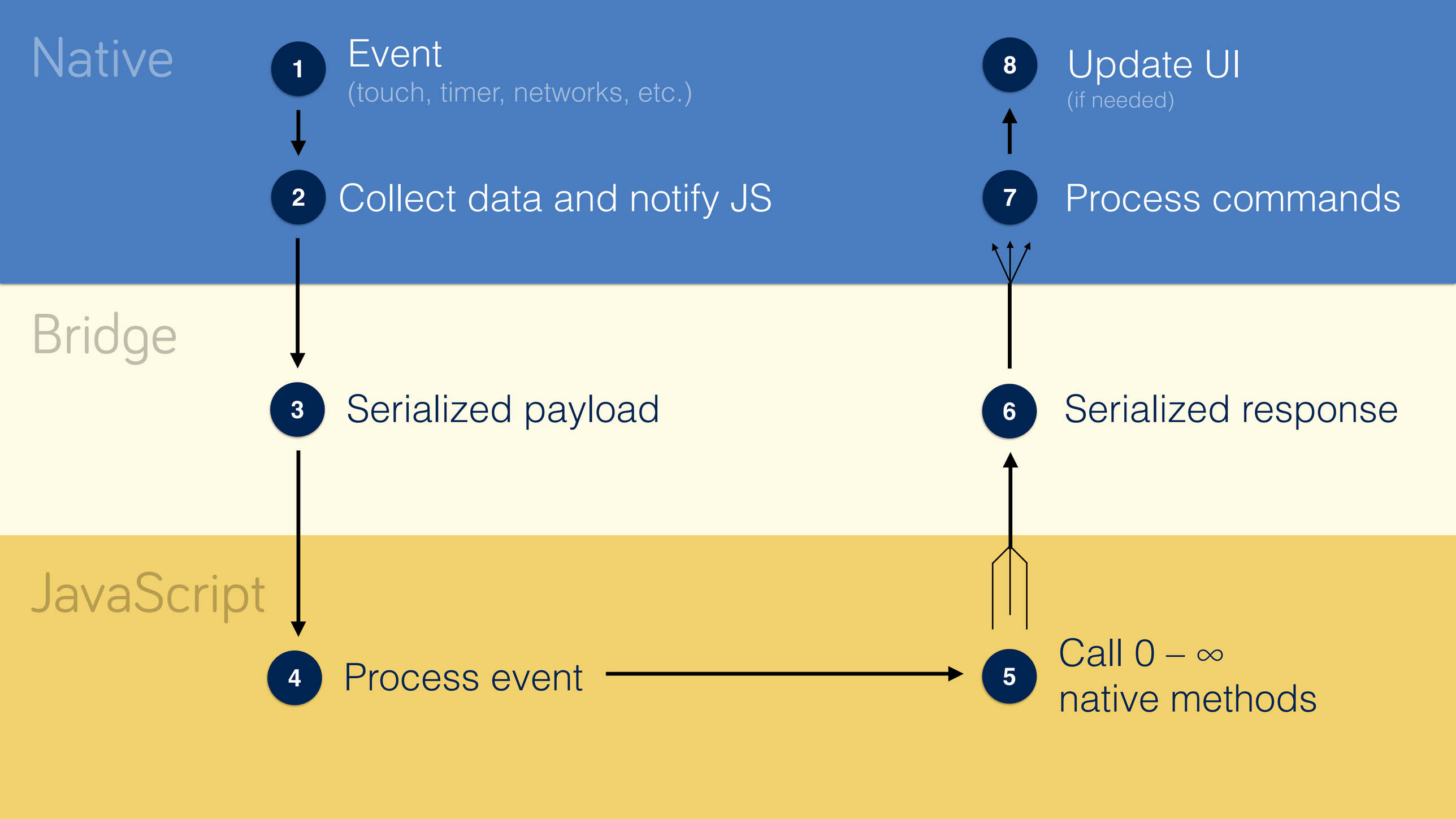
Base components

The Bridge

Native

Bridge

JavaScript

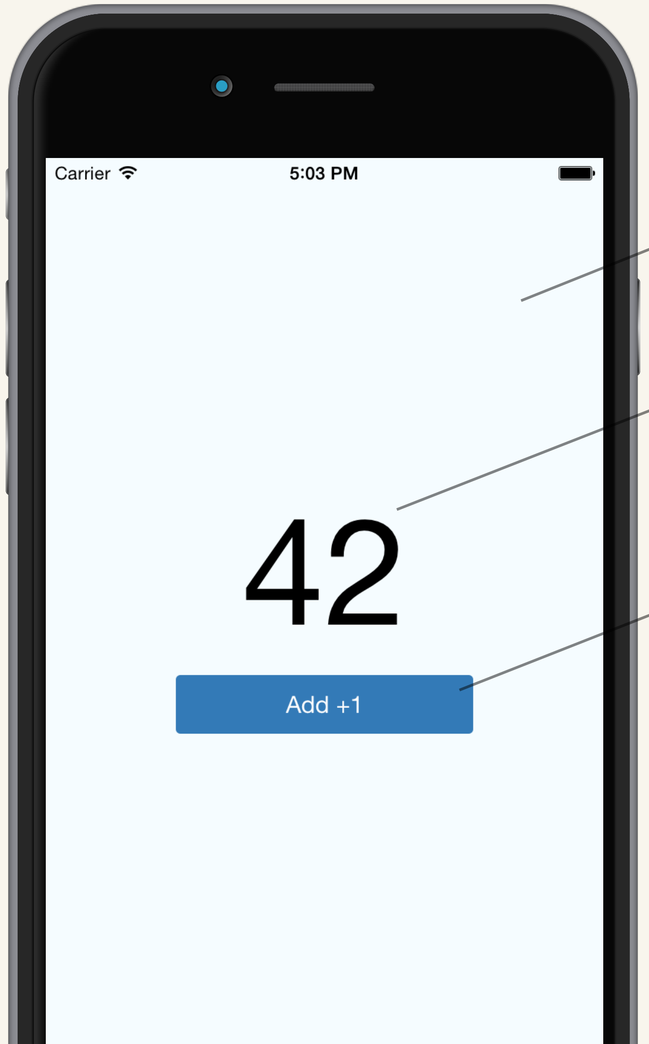

Native
1
Event
(touch, timer, networks, etc.)
2
Collect data and notify JS
8
Update UI
(if needed)
7
Process commands
Bridge
3
Serialized payload
6
Serialized response
JavaScript
4
Process event
5
Call 0 – ∞
native methods

JS is event-driven

Events →  → Commands

Example

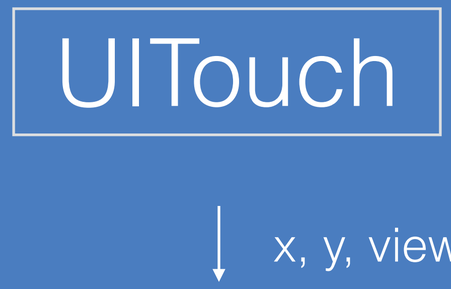
- Updates counter
- Sends data to web service




```
render() {
  return (
    <View style={styles.container}>
      <Text style={styles.value}>
        {this.state.count}
      </Text>
      <Button
        title="Add +1"
        onPress={() => this.inc()}
      />
    </View>
  );
}
```


```
inc() {
  var newCount = this.state.count + 1;
  this.setState({count: newCount});

  fetch(
    'https://api.conunter.io/',
    {
      method: 'post',
      body: 'value=' + newCount
    }
  );
}
```




```
[_bridge enqueueJSCall:@"EventEmitter.receiveTouches"
                  args:@[@"end",
                         @{@"x": @42, @"y": @106}]];
```

Native

Native

```
[_bridge enqueueJSCall:@"RCTEventEmitter.receiveTouches"
                  args:@[@"end",
                         @{@"x": @42, @"y": @106}]];
```

Bridge

```
[
  'EventEmitter', 'receiveTouches',
  ['end', {'x': 42, 'y': 106}]
]
```

JavaScript

```
call('EventEmitter', 'receiveTouches', [{x: 42, y: ...}])
```