React Native at Glow

Practices and Learnings

Allen, iOS Engineer, Glow Inc.

About Me

许帅

Allen

Tencent & Glow

iOS & Web Engineer

WeChat: imallen

Weibo: 许小帅_allen

About Glow

Glow - Fertility Tracker

Nurture - Pregnancy Tracker

Baby - Baby Tracker

Eve - Period Tracker

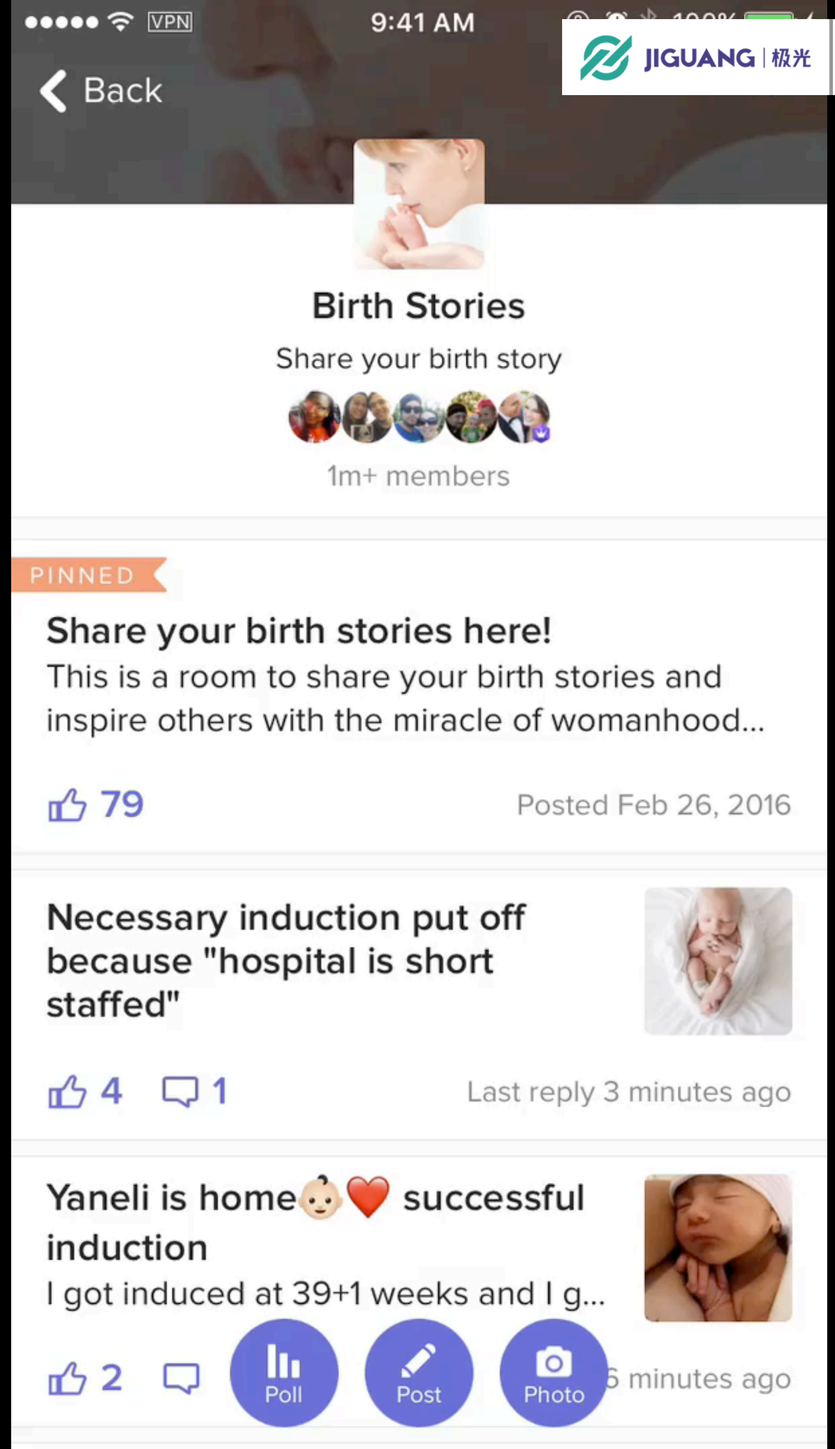
Glow Community

React Native

1.5 months

LOC (iOS): 46 K -> 14 K

Size (iOS): 6.6 MB -> 1 MB



Why React Native

Hot upgrade

Write once, run on both iOS and Android

Faster deployment, 8 apps -> 1 Jenkins job

Efficient workflow

"Performance"

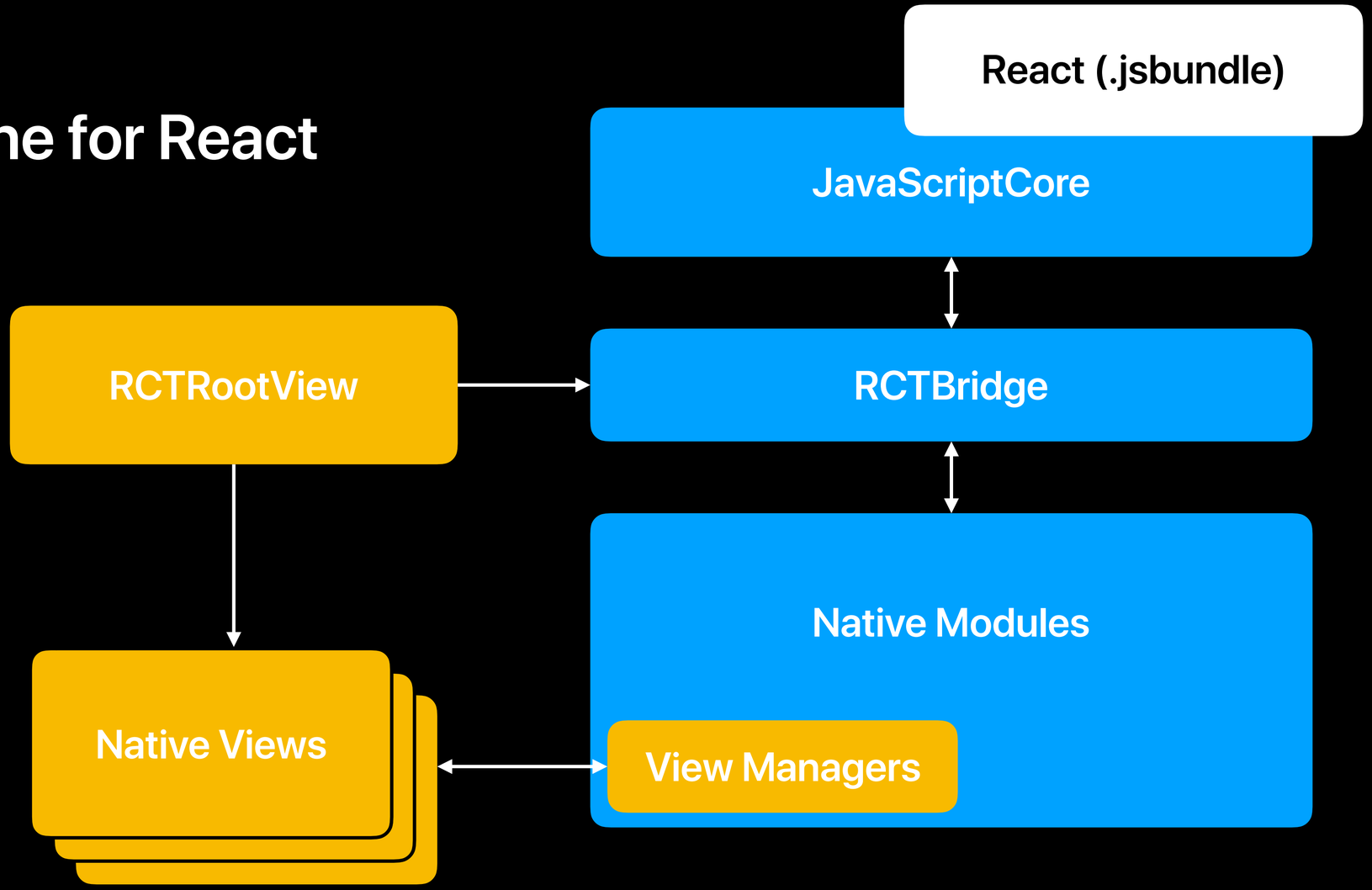
Cordova/PhoneGap

Weex

React Native

Native render engine for React

Native APIs



React
Component & JSX

```
class HelloWorld extends Component {
  render() {
    return (
      <View style={styles.container}>
        <Text numberOfLines={1}>Hello, {this.props.name}</Text>
      </View>
    )
  }
}
```

React
Component & JSX

```
<View style={styles.container}>
  <Text numberOfLines={1}>Hello, {this.props.name}</Text>
</View>

React.createElement(
  View,
  { style: styles.container },
  React.createElement(
    Text,
    { numberOfLines: 1 },
    "Hello, ",
    this.props.name
  )
)

React.createElement(component, props, ...children)
```

React
State & Props

```
class HelloState extends Component {
  componentDidMount() {
    this.state = { name: 'Unknown' }
    setTimeout(() => {
      this.setState({ name: 'Allen' })
    }, 3000)
  }
  render() {
    return (
      <HelloWorld name={this.state.name} />
    )
  }
}
```

URL Routing
& Native Navigation

URL Routing

Deep link

Decouple

Fallback

e.g. https://glowing.com/community/topic/1001

URL Routing

```
class RootComponent extends Component {
  render() {
    const { url } = this.props
    let match = null
    if (match = url.match(/\/community\/topic\/(\d+)/)) {
      return <TopicPage topicId={match[1]} />
    } else if (match = url.match(/\/community\/group\/(\d+)/)) {
      return <GroupPage groupId={match[1]} />
    } else {
      return <Text>404 Page Not Found</Text>
    }
  }
}

AppRegistry.registerComponent('main', () => RootComponent)
```

URL Routing

```
- (RCTRootView *)rootViewForURLString:(NSString *)urlString
                                props:(NSDictionary *)props
{
    RCTBridge *bridge = self.sharedBridge;
    if (!bridge) return nil;

    NSMutableDictionary *initialProperties =
        [NSMutableDictionary dictionaryWithDictionary:props];
    if (urlString) initialProperties[@"url"] = urlString;

    return [[RCTRootView alloc] initWithBridge:bridge
                                    moduleName:@"main"
                             initialProperties:props];
}
```

URL Routing

```
[[GLRNManager sharedManager] rootViewForURLString:@"/community/topic/1001"
props:nil];
```

```
<RootComponent url="/community/topic/1001" />
```

```
<TopicPage topicId="1001" />
```

URL Routing

```
const routeMap = {
  '/community/topic/:topic_id': (url, params, props) => {
    return <TopicPage topicId={params.topic_id} {...props} />
  },
}

const components = this.urlComponentsFromUrl(this.props.url)
const queryParams = this.queryParamsFromString(components.query)

for (var pattern in routeMap) {
  const params = this.matchParams(components.path, pattern)
  if (params !== null) {
    const renderFunc = routeMap[pattern]
    return renderFunc(url, {...queryParams, ...params}, this.props)
  }
}
```

Native Navigation

```
<TouchableHighlight onPress={() => {
  NativeNavigator.showURL(`/community/topic/${topic.id}`)
}}>
  <Text>Topic Link</Text>
</TouchableHighlight>
```

Immutable

Prevent Unexpected Side Effect

Immutable
The Problem

```
var a = { name: 'Allen' }
var b = a

b.name = 'Nella'

console.log(a) // { name: "Nella" }
a === b // true
```

Immutable
Object Spread

```
var a = { name: 'Allen' }
var b = { ...a }

a === b // false

b.name = 'Nella'

console.log(a) // { name: "Allen" }
a === b // false
```

Immutable
Nested Object

```
var a = { author: { name: 'Allen' } }
var b = { ...a }

b.author.name = 'Nella'

console.log(a) // { author: { name: "Nella" } }
a.author === b.author // true
```

Immutable

```
const { fromJS } = Immutable

const a = fromJS({
  author: {
    name: 'Allen',
  },
  replier: {
    name: 'Grace',
  },
}) // Immutable.Map

let b = a.setIn(['author', 'name'], 'Allen')
console.log(a === b) // true
b = b.setIn(['author', 'name'], 'Nella')
console.log(a === b) // false
console.log(a.get('author') === b.get('author')) // false
console.log(a.get('replier') === b.get('replier')) // true
```

Immutable

Better performance with React.PureComponent

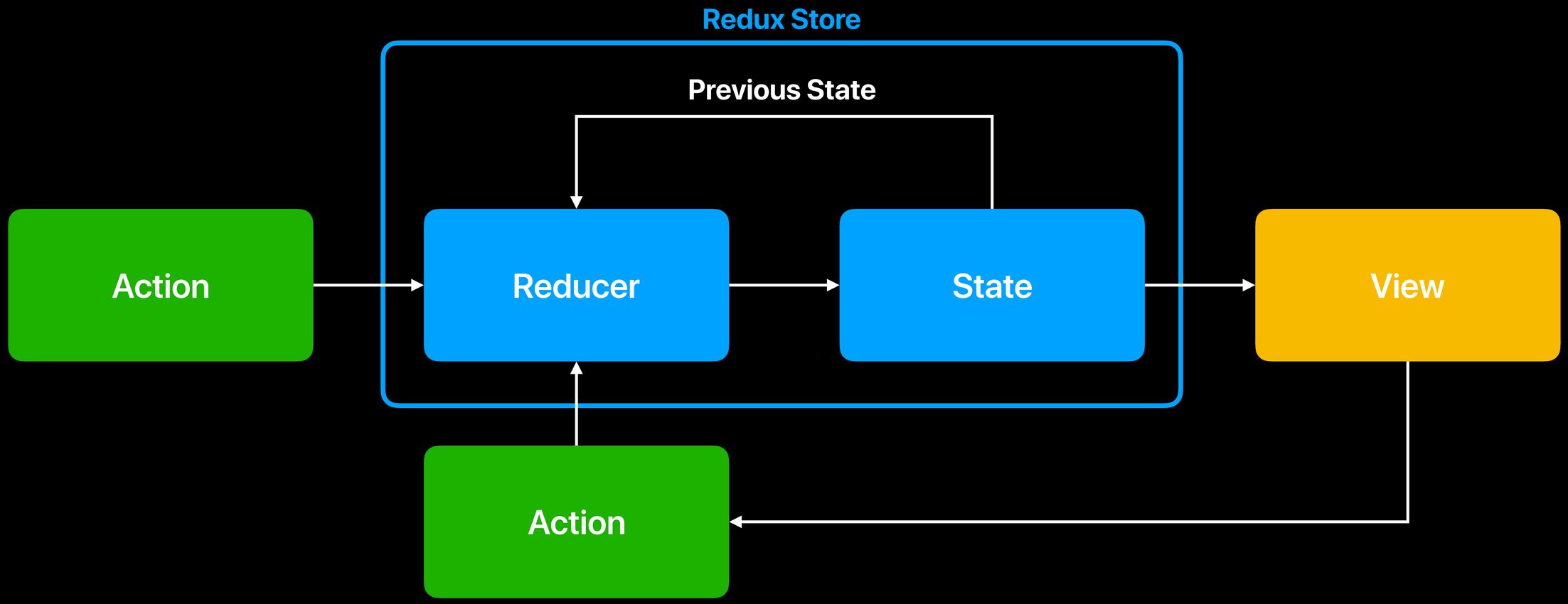
```
// Component
shouldComponentUpdate(nextProps, nextState) {
  return true
}

// PureComponent
shouldComponentUpdate(nextProps, nextState) {
  return !shallowEqual(prevProps, nextProps) ||
    !shallowEqual(prevState, nextState)
}
```

Get prepared for multi-threading

Prevent unexpected side effect

Redux

Redux


Redux Store
Previous State
Action
Reducer
State
View
Action

React-Redux

```
import { connect } from 'react-redux'

const mapStateToProps = (state) => ({
  todos: state.todos,
})

class App extends Component {
}

export default connect(mapStateToProps)(App)
```

React-Redux

Action:

```
{ type: 'ADD_TODO', payload: "New item" }
```

Reducer:

```
const todoReducer = (state, action) {
  switch (action.type) {
    case 'ADD_TODO':
      return {
        ...state,
        todos: [action.payload, ...state.todos]
      }
  }
  return state
}
```

Data Normalization

Single source of truth

Data Normalization

```
{
  "id": "1001",
  "author": {
    "id": "1",
    "name": "Allen"
  },
  "title": "Topic title",
  "content": "Topic content",
  "comments": [
    {
      "id": "123",
      "content": "Comment content",
      "author": {
        "id": "1",
        "name": "Allen"
      }
    }
  ]
}
```

Data Normalization

```
import { normalize, schema } from 'normalizr'

const user = new schema.Entity('users')

const comment = new schema.Entity('comments', {
  author: user
})

const topic = new schema.Entity('topics', {
  author: user,
  comments: [ comment ]
})

const normalizedData = normalize(originalData, topic)
```

Data Normalization

```
{
  entities: {
    users: {
      '1': { id: '1', name: 'Allen'}
    },
    comments: {
      '123': { id: '123', content: 'Comment content', author: '1' }
    },
    topics: {
      '1001': {
        id: '1001', author: '1',
        title: 'Topic Title', content: 'Topic content',
        comments: ['123']
      }
    }
  },
  result: '1001'
}
```

Data Normalization

```
import { normalize, denormalize, schema } from 'normalizr'

const user = new schema.Entity('users')

const comment = new schema.Entity('comments', {
  author: user,
})

const topic = new schema.Entity('topics', {
  author: user,
  comments: [ comment ],
})

const normalizedData = normalize(originalData, topic)
const { entities } = normalizedData
const myTopic = denormalize(1001, topic, entities)
```

Q&A