React.js in Strikingly

Dafeng Guo (郭达峰), CTO of Strikingly

strikingly

What I won’t talk about

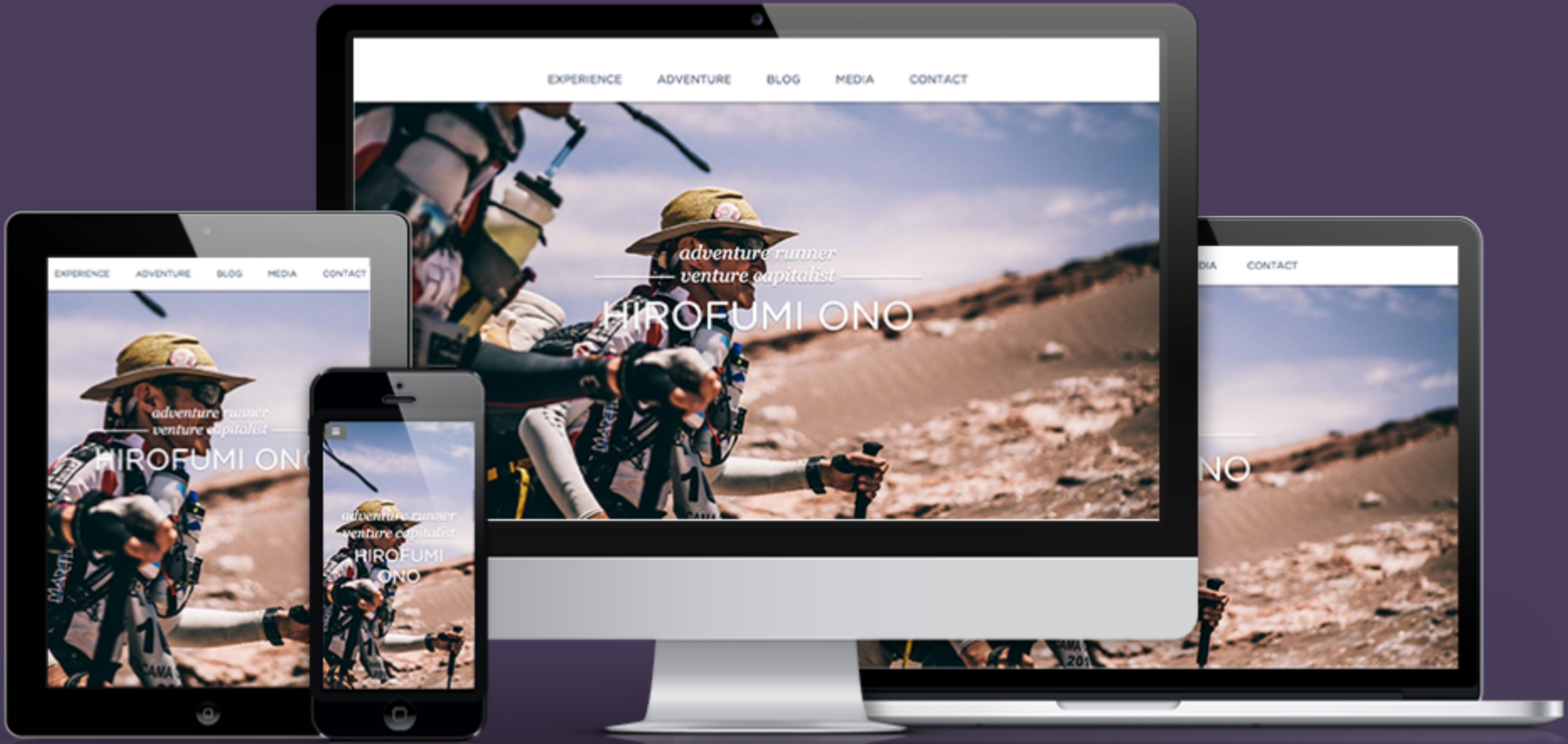
- How to write React.js

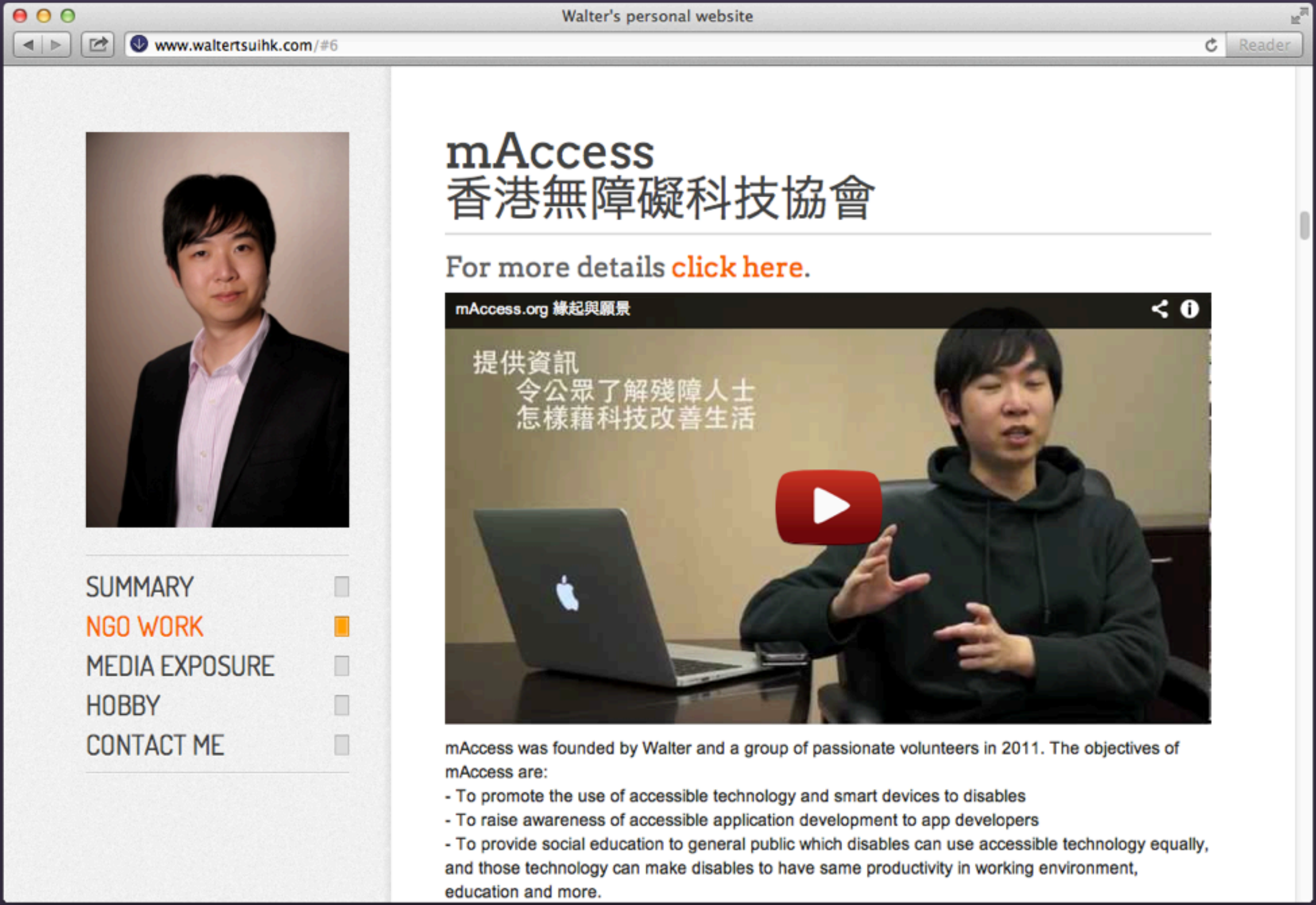
What I will talk about?

- A lot of new things are introduced in the frontend world
- Why React.js is revolutionary and you should know it
- 目标：听完分享后你会认同

Who am I?

- 郭达峰 (dfguo)
  - Have been hacking since 14
  - CTO of Strikingly


EXPERIENCE
ADVENTURE
BLOG
MEDIA
CONTACT
adventure runner
venture capitalist
HIROFUMI ONO



mAccess
香港無障礙科技協會

For more details click here.

mAccess was founded by Walter and a group of passionate volunteers in 2011. The objectives of mAccess are:
- To promote the use of accessible technology and smart devices to disables
- To raise awareness of accessible application development to app developers
- To provide social education to general public which disables can use accessible technology equally, and those technology can make disables to have same productivity in working environment, education and more.

Y Combinator

W13 Class

Why Strikingly chose React.js?

- We have a pretty complex frontend
  - Demo
- Existing front-end tech stack:
  - Angular.js + Knockout.js
- We still have a lot of issues with performance and code organisation
- Can React help? Let's see.

State management problem

- Server-side rendering
  - Rails, Django, php
  - Fresh state every time but not interactive

- Client-side two-way binding
  - Angular.js, Ember.js, Knockout
  - Interactive but complex to manage the states

Why is it hard to manage the states?

- We always juggle between changing state and then changing DOM
- State and DOM changes over time make it hard to synchronize

How do we make DOM the exact reflection of states?

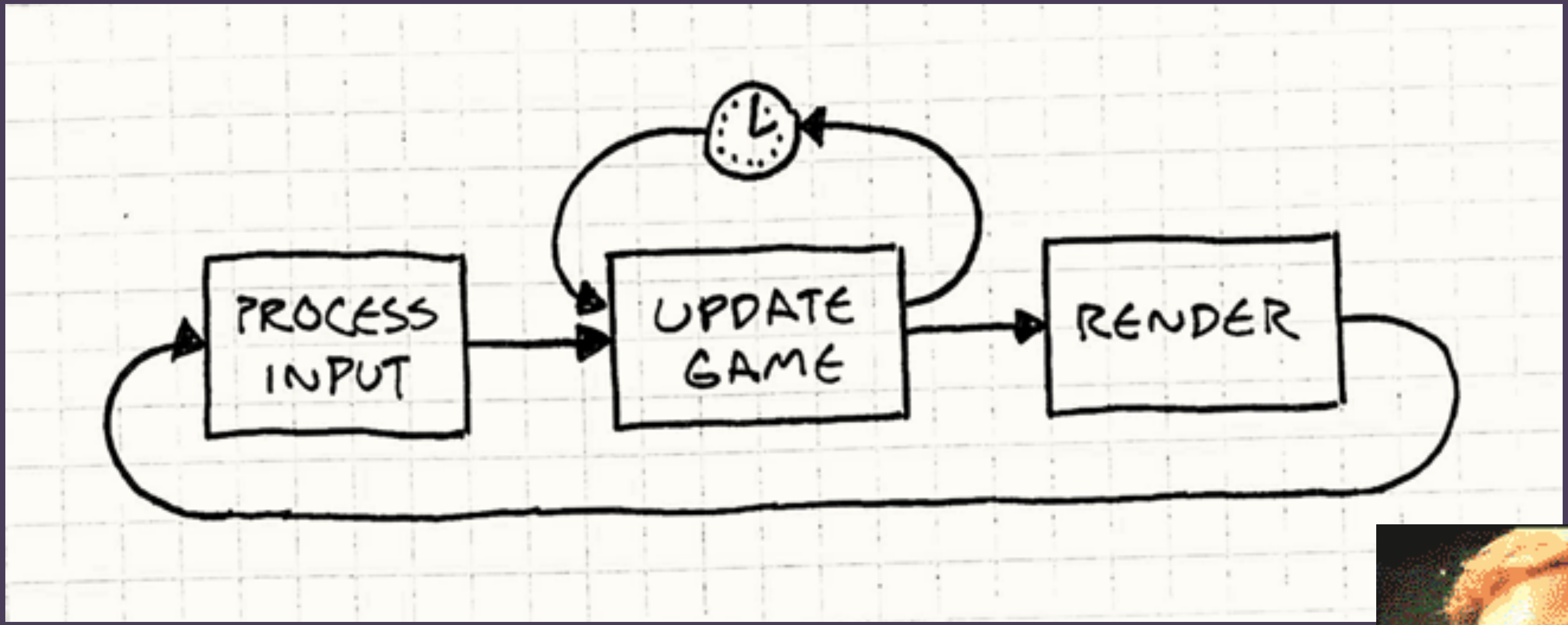
- What if we can re-render the DOM all the time, like backend rendering?
    - state tracking will not be a problem
    - but it’s horribly slow

Virtual DOM

- Virtual DOM
    - Keep a copy of the DOM in javascript memory
    - When DOM needs to change, get the diff by comparing virtual DOM
    - Update the diff to the real DOM

Virtual DOM

Just like game rendering



Virtual DOM

- It’s fast!
  - Only update the diff
  - Batch update
  - Faster than any other frameworks

Virtual DOM

- Sample code time: https://facebook.github.io/react/jsx-compiler.html
- Still hate it? Check out React-template

Pre-rendering

- because DOM can be generated with javascript runtime, you can render React.js from backend
- Isomorphic javascript - run the same javascript from backend and frontend

Pre-rendering

- This is important to Strikingly for SEO and speed
  - We used phantomJS to simulate a browser and generate the static HTML in the old architecture
  - With React.js, we can use Rails + execjs + node.js to render

Testing

- SLOW - Most annoying thing about integration test or any test that requires browser
- FAST - React.js tests just need javascript runtime

- Faster DOM changes
- Server-side rendering out of the box
- Faster testing

virtual DOM enables all these

- But virtual DOM is not everything. I talked about it because it’s easier to understand.
- The core of React.js is to to allow:
    - **UI = f(states)**
    - write declarative code that manages application state and DOM together!!

Unidirectional Data Flow

- UI as state machine
  - Given data, display UI: f(states) = UI
- Not two-way, but one-way
- Data flow from top to bottom, parent to child
  - Flux to facilitate this flow
  - Immutable.js makes it even better

Unidirectional Data Flow

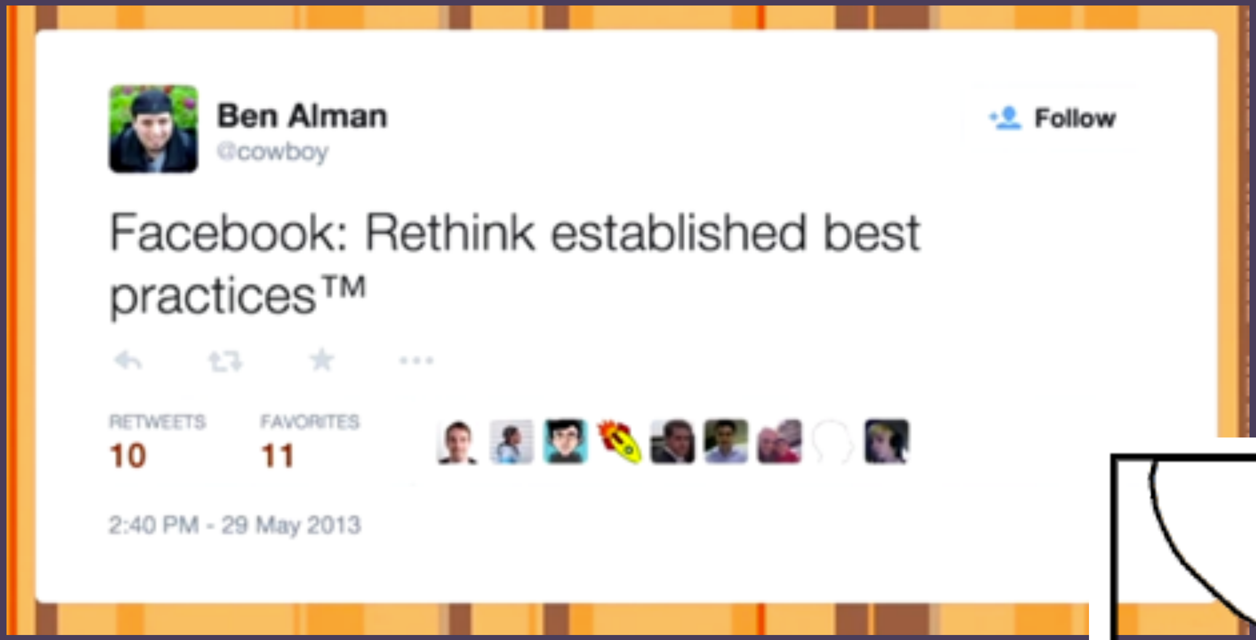
- Functional programming: pure function
  - idempotent - f(state) is always y
  - composable - 可组合性

Unidirectional data flow

Virtual DOM

They are just tools to help us to write **f(states) = UI** and not worry about manipulating both states and DOM separately

Conclusion

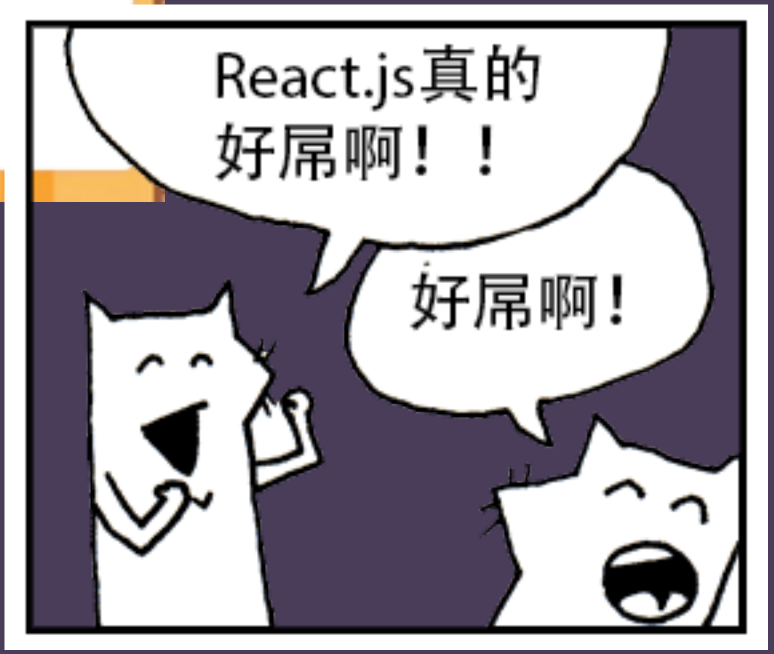

Ben Alman
@cowboy

Follow

Facebook: Rethink established best practices™

RETWEETS	FAVORITES
10	11

2:40 PM - 29 May 2013



小广告

Strikingly 正在招聘

React China

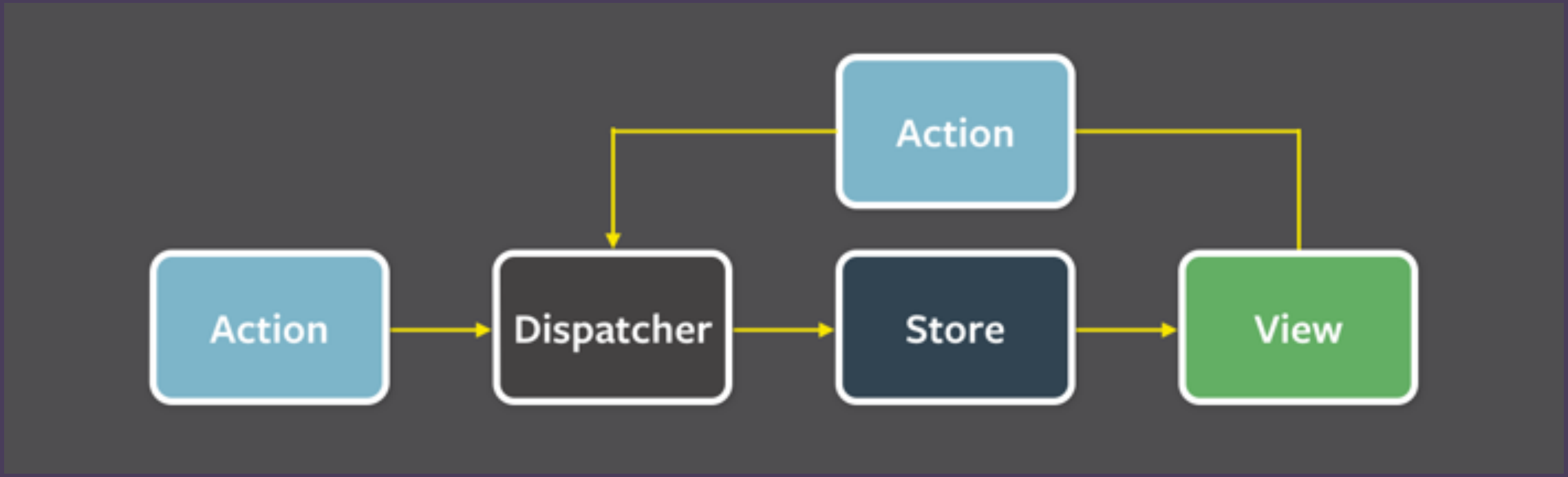
References

- Pete Hunt's Rethinking Best Practices: https://www.youtube.com/watch?v=DgVS-zXgMTk
- Immutability and React: https://www.youtube.com/watch?v=I7IdS-PbEgI

- Additional material if audience wants to learn more about Flux and immutable.js

Flux

- Unidirectional data flow



Immutable.js

- Immutable - changes create new copy on every change
- Persistent - old copies will be kept
- Structural sharing - new/old objects shares memory

Immutable.js

- Makes one-way directional flow clearer
- Makes state changes more clear
- undo/redo out of the box :D