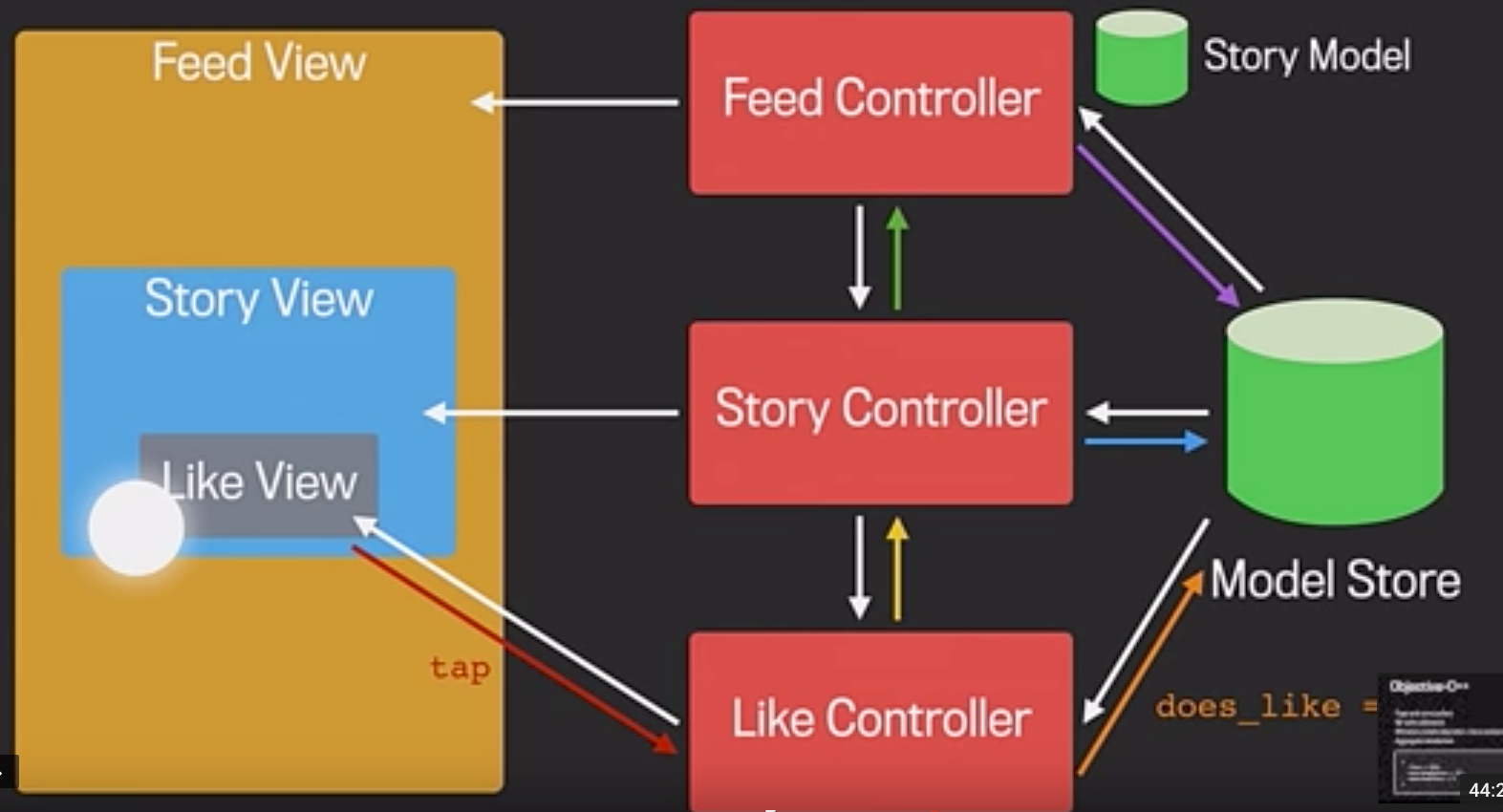
**ComponentKit**

1. **Vấn đề**

Các vấn đề gặp phải khi tạo UI trên IOS

* Phải viết rất nhiều imperative code
* Phải implement callback cho controller( có thể làm thay đổi trạng thái của view tạo thành data flow 2 chiều). Data flow 2 chiều làm cho việc debug và maintain trở nên khó khăn hơn đặc biệt là trong môi trường multithread 

ComponentKit được Facebook tạo ra để giải quyết những khó khăn trên. Facebook đã sử dụng ComponentKit để tạo New Feed UI và kết quả đạt được:

* Giảm 70% số lượng code để tạo UI
* Bỏ hết manual layout code
* Cải thiện Scroll performance
* Cải thiện test coverage. ComponentKit giúp cho developer có thể tạo UI thành từng module nhỏ và có thể test độc lập.

1. **Giới thiệu ComponentKit**

ComponentKit là Objective-C++ view framework cho IOS và được truyền cảm hứng từ React. Mục tiêu của ComponentKit là:

* Dễ layout và recycle views
* Data flow 1 chiều khi state thay đổi
* Khuyến khích **composition** và bắt buộc **imutablity**
* Làm giảm thiếu lỗi
* Allow asynchronous computation **invisibly (text layout, image decode)**

ComponentKit sử dụng cách tiếp cận Declarative, Functional để tạo UI và tập trung vào data flow 1 chiều từ immutable models -> immutable compoents để cách mô tả cách view được configured.

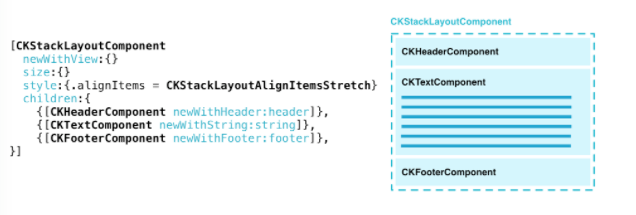


Cách tiếp cận declarative của ComponentKit giúp cho developer tập trung vào UI nên được hiển thị như thế nào, thay vì tập trung vào từng bước để tạo ra UI như cách tiếp cận imperative truyền thống.

Ví dụ để hiển thị 1 view có header, text và footer bằng cách thông thường:

* Tạo header view, text view, fotter view
* Thêm 3 view trên thành subviews của view của tạo
* Thêm constraints cho từng view để 3 view có width bằng với container
* Thêm constraints để layout vị trí chính xác của từng view

Khi làm việc với ComponentKit, nó cần developer mô tả \*what you want\* thay vì \*how it should be done\* như cách trên. Ví dụ: "Tôi muốn 1 header component, 1 text component và 1 footer component stack với nhau theo chiều vertical và giãn ra theo chiều horizontal.



ComponentKit abstracts việc render UI nên giờ developer chỉ cần mô tả Component còn việc tạo UI như thế nào thì để ComponentKit lo.

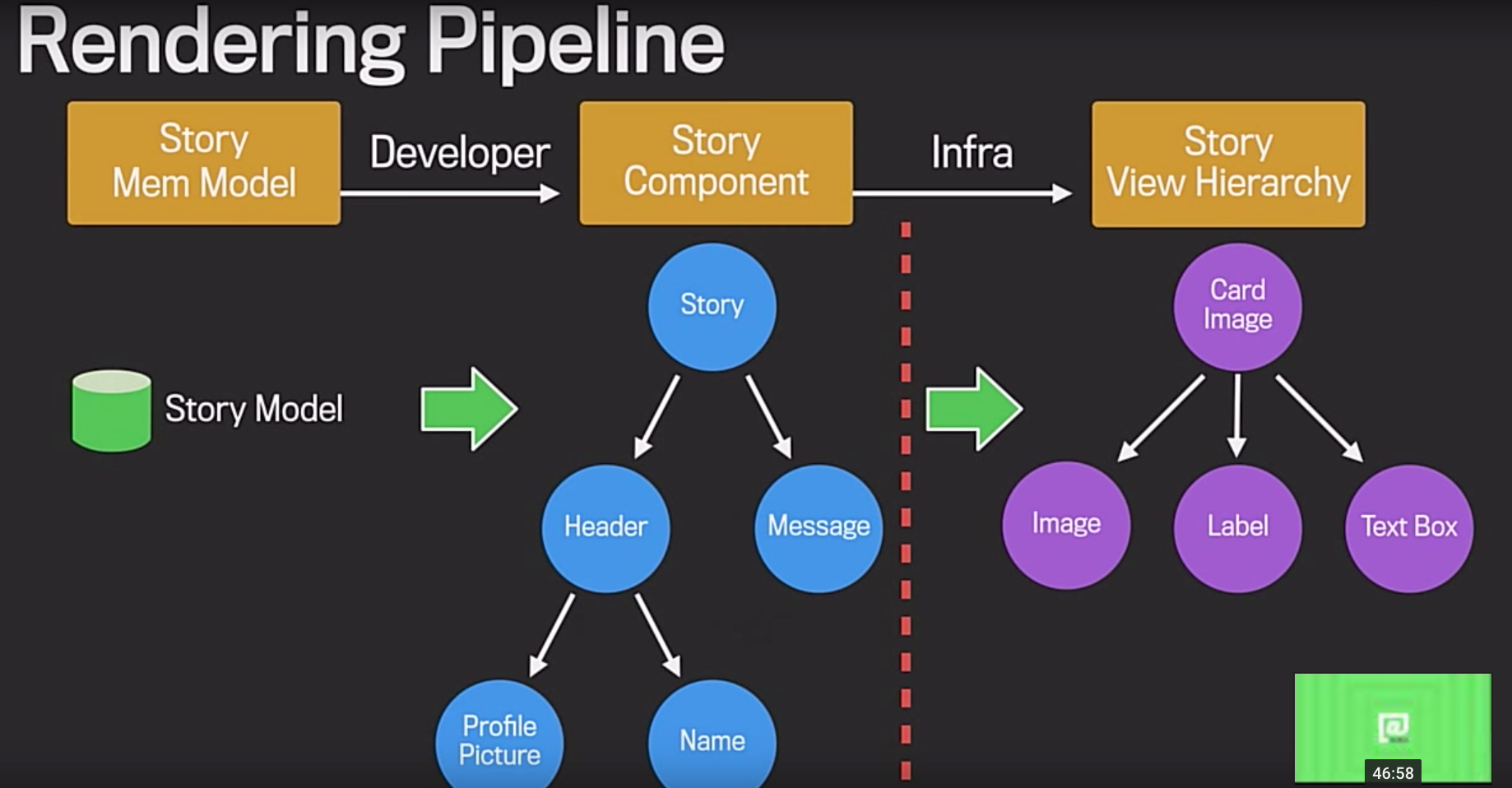
ComponentKit thực hiện layout trên background thread và tạo view hierachy nhỏ nhất để rander components lên screen giúp cho scroll performance mượt hơn.

1. **Kiến trúc và cách hoạt động**

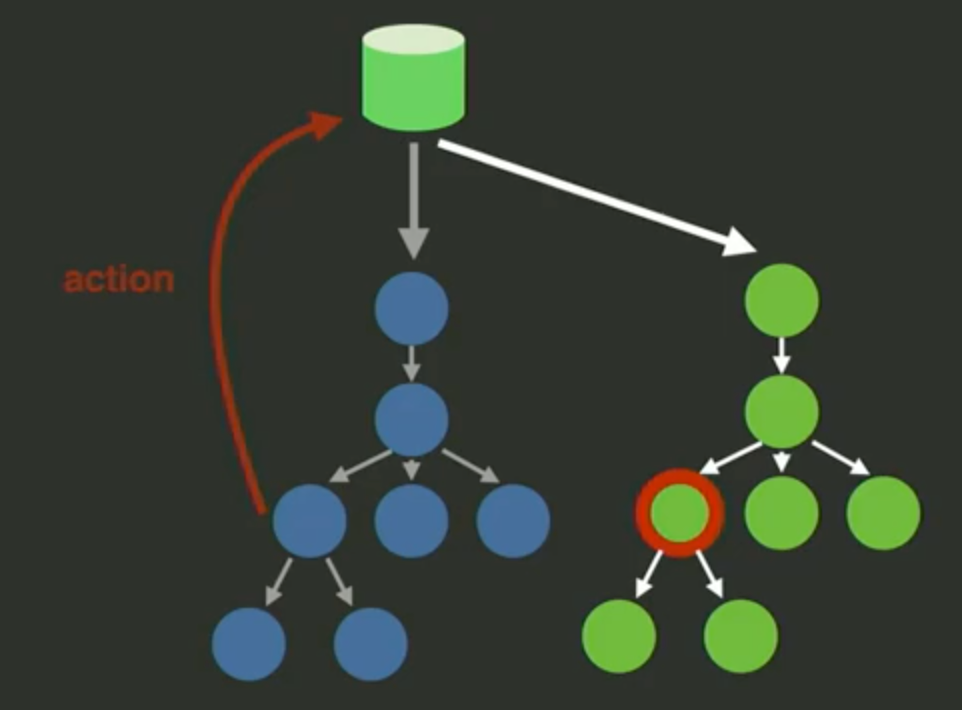
**Component**

Trong ComponentKit, đối tượng chính mà chúng ta cần xử lý là Component. Component nhận input là 1 immutable model và return 1 immutable components tree. Rồi sau đó ComponentKit sẽ tự chuyển từ components tree thành hirachy UIVIew. Tất cả các quá trình đều được thực hiện trên background thread.

Mỗi component có thể được render như 1 stack ( để layout các component con) hoặc render thành các UIView. Component là immutable do đó không có hàm addSubComponent, khi cần thêm component thì ta phải tạo component mới và thêm ở hàm khởi tạo.



Khi có sự thay đổi, ComponentKit sẽ tạo 1 component tree mới, chạy thuật toán diffing trên background thread để tìm component vừa được thay đổi. Thuật toán sẽ duyệt tree theo hướng Top-Down, khi duyệt tới component cần thay đổi thì ComponentKit chỉ thay đổi component(sub components) đó thành component mới.



**Layout Components**

ComponentKit tạo sẵn 1 số components hỗ trợ layout

* CLStackLayoutComponent : layout tương tự CSS flexbox, cho phép các components stack vertically hoặc horizontally và cách các components aligned trong stack
* CKInsetComponent : tạo 1 inset margin xung quanh component
* CKCenterLayoutComponent
* CKRatioLayoutComponent
* CKStaticLayoutComponent
* CKBackgroundLayoutComponent
* CKOverlayLayoutComponent

Ngoài ra bạn có thể tự tạo layout bằng cách implement computeLayoutThatFits:

Xem thêm http://componentkit.org/docs/layout.html

**State**

ComponentKit sử dụng khái niệm props và state tương tự như React. Mỗi component gồm có 2 elements:

* props: Không thể thay đổi ở trong component. Props được truyền từ component cha , props trong ComponentKit chính là tham số được truyền ở hàm +new
* state: Trạng thái có thể bị thay đổi bên trong bản thân component nhưng không bị thay đổi bới các component khác

Khi props và state đều trigger render update khi bị thay đổi

Khởi tạo và thay đổi state trong CKComponent bằng cách hàm sau

**@interface** **CKComponent**

**-** (**void**)**updateState**:(id (^)(id))updateBlock **mode**:(CKUpdateMode)mode;

**+** (id)initialState

**@end**

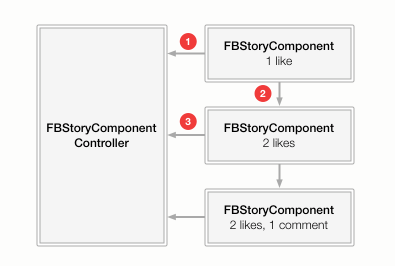
**Component Controller**

Component có chu trình sống ngắn(short-lived) do thường xuyên bị xóa và tạo component khi có sự thay đổi và chúng ta không thể kiểm soát kiểm soát được life cycle của components.

Đôi khi ta cần 1 object có chu kỳ sống dài hơn để xử lý ví dụ như:

* Components không thể implement Delegates do thường xuyên bị xóa tạo lại
* Network download. component cũ có thể đã bị xóa khị download xong

Component Controller được tạo giải quyết trên, tạo controller bằng cách tạo subclass của CKComponentController có tên trùng với component và thêm suffix "Controller".



Mặc dù component bị xóa tạo lại nhưng vấn reference tới chung 1 Controller

**Truyền dữ liệu giữa Component và Component Controller**

Chỉ có thể truyền dữ liệu từ Component sang Component Controller mà không thể ngược lại. Tạo @property trong component, controller có thể nhận giá trị thông qua hàm **initWithComponent:** .

Nếu property có thể bị thay đổi khi state bị thay đổi ( giá trị property muốn truyền khác nhau giữa các lần tạo component) thì ta có thể sử dụng hàm **didUpdateComponent** để bắt sự kiện change

Lưu ý: Component không có property giữ reference tới Controller mà ta phải tạo Controller có tên trùng với Component và thêm suffix "Controller"

Xem thêm ví dụ: http://componentkit.org/docs/component-controllers.html

1. **Ưu và nhược điểm**

* Ưu điểm
* Simple và Declarative
* Scroll Performance: Tất cả layout được thực hiện trên background đảm bảo 60 FPS
* View Recycling: Tất cả configruation đều được khai báo trong component( declarative và immutable) giúp hạn chế lỗi khi view recycling
* Composability: Khuyến khích sử ụng composition giúp hạn chế số lượng code cho từng component
* Nhược điểm
* Không phù hợp với những interface không phải là lists
* Build bằng Objective-C++ nên không hợp tích hợp với Swift
* Khó tạo animate phức tạp
* Khó implement dynamic gesture-driven UI

Kết luận: ComponentKit chỉ phù hợp để xây dựng những UI tĩnh và layout phức tạp như Feed

1. **Tài liệu tham khảo**

* https://code.facebook.com/posts/1415586422080360/introducing-componentkit-functional-and-declarative-ui-on-ios/
* https://www.netguru.co/blog/imperative-vs-declarative
* http://componentkit.org/
* https://news.realm.io/news/pragma-hannes-verlinde-statelessness-react-native/
* https://news.realm.io/news/altconf-benjamin-loulier-reactive-ui-on-ios-just-tell-me-what-not-how/