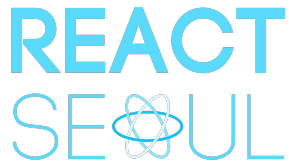


# React/Redux for Smart TV UI

2017. 11. 4

[dongyoung.lee@lge.com](mailto:dongyoung.lee@lge.com)

[dongyoung.lee@gmail.com](mailto:dongyoung.lee@gmail.com)



# 이 발표의 개요

스마트 TV UI에 React/Redux를 적용한 사례

Embedded 환경의 특성

React/Redux 적용을 통해 얻게 된 장점 및 교훈

성능 이슈

# LG webOS TV

“web”OS TV: web application이 많다

기존 framework: Enyo (<http://enyojs.com/>)

성능, 유지보수, 최신 기술 적용 어려움

The screenshot shows the Enyo JavaScript Application Framework website. The browser address bar displays "Enyo JavaScript Ap x" and "enyojs.com". The navigation menu includes "About", "Get Enyo", "Docs", "Showcase", and "Community". The main heading is "Meet Enyo", followed by the tagline "a framework for building native-quality HTML5 apps that run everywhere". A prominent green button says "Get Enyo Now". Below this, three key features are highlighted: "Cross-Platform" (with a monitor icon), "Battle Tested" (with a flame icon), and "100% Free" (with a globe icon). Each feature has a brief description. At the bottom, there is a section for "Enyo 2.7.0" with a description of the new module system and a small image of a user interface.

Enyo JavaScript Ap x  
enyojs.com

ENYO About Get Enyo Docs Showcase Community

Meet Enyo  
a framework for building native-quality HTML5 apps that run everywhere

Get Enyo Now

**Cross-Platform**  
Use Enyo to develop apps for all major platforms, from phones and tablets to PCs and TVs

**Battle Tested**  
Enyo has powered apps for mobile devices, desktops and LG webOS Smart TVs

**100% Free**  
Available under the Apache 2 License, Enyo is open-source and completely free to use

**Enyo 2.7.0**  
Enyo 2.7.0 is now available. The most significant feature in Enyo 2.7 is a shift to a CommonJS-style module system. The new system lets Enyo developers create highly optimized

# 신규 Framework 선택의 기준 (Polymer vs. React)

기능 요구사항 만족

성능, 기술적 매력 (Extensible Web)

유지보수, 성숙 정도, 생태계

앱 구조 제공 (Redux)

Netflix

# Embedded 환경의 특성 (1)

App이 device에 저장되어 있다. (Loading 특성, 업데이트, SEO)

API (HTTP vs. system bus)

CPU 느리다.

서비스도 느리다. (서버의 도움 받을 수 없다.)

Disk write 함부로 하면 안된다.

Peak 메모리 사용량 중요하다.

## Embedded 환경의 특성 (2)

특정 Hardware들/OS/웹 엔진에서만 실행된다.

화면 크기/비율 고정

Polyfill

웹 엔진을 고칠 수 있다. OS도 고칠 수 있다.

Preloading

Scroll

## Embedded 환경의 특성 (3)

Server-side rendering (at runtime) vs. Prerendering (at build time)

Static HTML을 미리 만들어서 launch 시간 단축

많은 것들이 runtime에 결정된다. (예: 언어)

# TV Applications

5-way navigation 지원 (상하좌우 + OK)

+ LG 매직 리모컨 (마우스)

Audio guidance

Internationalization: 번역, RTL, Layout, 줄바꿈 규칙, 폰트, 시간/날짜 포맷, ...



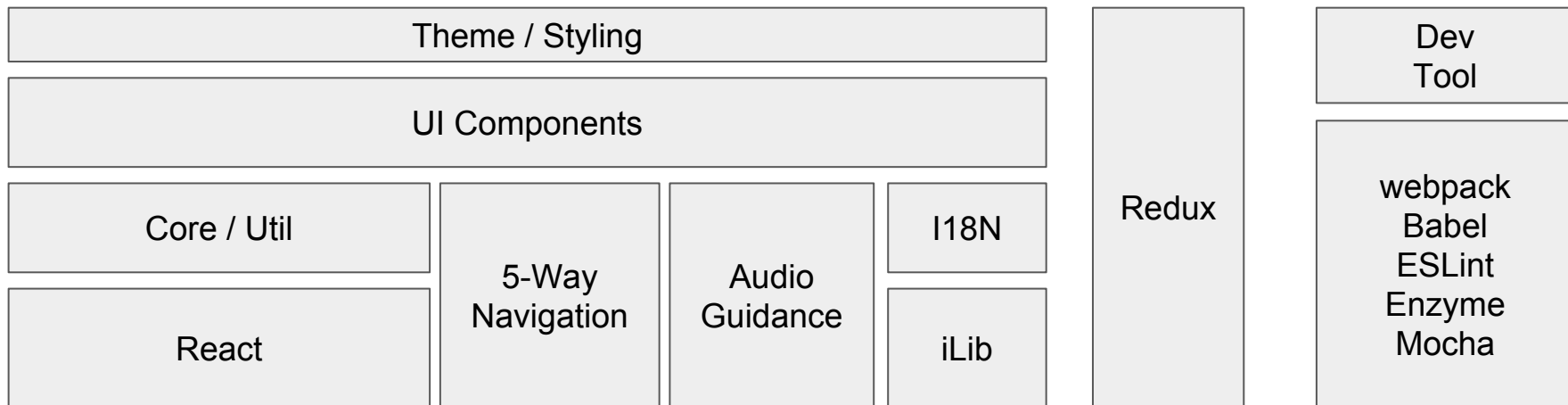
iLib-js <https://github.com/iLib-js>



<https://www.transifex.com/blog/2015/localization-friendly-ui-design-part-1/>



# TV Web App Framework



# Migration

UI Components, TV 특화 기능 (5-way, accessibility, I18N)

20+ Applications

2년에 걸쳐서

성능 개선이 필요한 App, 새로 개발하는 App, Framework 완성도, 개발자 업무량

# 성능 / 메모리 비교

앱	구분	Old	New	%
S	Launch	1.74	1.5	116
	전환	1.43	1.07	134
R	Launch	6.43	2.31	278
P	Launch	2.13	1.75	121
	전환	1.03	0.87	118
	FPS	30	60	200
C	Launch	1.5	1.37	109
	전환	6.75	1.9	355
M	Launch	1.87	1.81	103
	전환	1.6	1.66	96
B	Launch	1.74	1.18	147
N	Launch	2.46	2.2	111
평균				152

앱	Launch 후			Peak		
	Old	New	%	Old	New	%
S	75	77	103	99	113	113
R	36	49	133	450	117	26
P	165	159	96	234	222	95
C	105	136	129	187	174	93
M	95	86	91	155	106	68
B	110	57	51	116	59	51
N	85	111	130	106	122	115
평균			105			80

# 품질/테스트/디버깅

ESLint

editor integration, build-tool integration

unit test (enzyme, mocha)

functional programming

react-devtools

redux-devtools

# 성능 요소 Overview

## 1. Redux

- Action 발생
- Store 업데이트 (Reducer)
- mapStateToProps

## 2. React

- Lifecycle hooks
- Render 함수
- reconciliation

## 3. Web Engine (DOM update)

- Style recalculation
- Layout
- Painting
- Compositing

# Redux 최적화

Action 발생 자체를 최소화해야 함

애니메이션 중 action 발생 X, Redux/React 우회 (ref)

성과와 품질(predictability, testability) 사이의 trade off

Store 업데이트 (Reducer) 시 immutability 위한 data 구조 고려 (예: 배열?)

mapStateToProps 함수는 간단해야 함 (과정, 결과)

# React 최적화

Lifecycle hooks 많으면 느리다 (예: 다단계 HOC)

Render 함수 간단하게

Render 횟수 최소화: react-addons-perf 꼭 사용하자

reconciliation (예: key 사용)

# 회고 및 교훈

생태계 중요하다. (학습 자료, 개발 도구, 업데이트)

비교적 배우기 쉬웠다.

성능보다는 품질 측면의 장점이 더 커 보인다. (functional programming)

성능 최적화 위해서 React/Redux에 대한 이해가 중요하다.

Redux Store/Action 설계 중요하다.

Render 횟수 최소화해야 한다.

shouldComponentUpdate 위험하다.

HOC가 너무 많이 중첩되면 느리다.



# 미래

React 16 (Fiber): priority scheduling

아직 성능 개선의 여지가 많다. (inferno)

Framework as an optimizing compiler. (svelte)