HTML5 기반 스마트 TV 플랫폼 표준 앱 검증 도구 개발[1]

황희선, 김호년, 이동훈, 박동영, 이은향
한국정보통신기술협회
{ domich.hwang, dhlee, hykimfnd, dypark, ehlee }@tta.or.kr

Implementation of Standard Conformance Verification Tool for Smart TV Platform based on HTML5

Hee-Seon Hwang, Dong-Hoon Lee, Ho-Youn Kim, Dong-Young Park, Eun-Hyang Lee
Telecommunications Technology Association

요 약

본 논문은 TTA 에서 제정된 “HTML5 기반 스마트 TV 플랫폼” 표준(TTAK.KO-07.0111/R1)에 따라 개발되는 스마트 TV 수신기 앱이 표준의 기술 요구사항을 준수하여 적합하게 구현되고 있는지 검증할 수 있는 앱 검증 도구를 소개한다. 이 도구는 개발자에게 제공되는 통합개발환경에 포함되는 개발 툴의 하나로, 표준기반 스마트 TV 앱의 소스코드를 해석하고 의미를 분석하여, 표준의 기술 요구사항에 따라 정의된 검증 규격(Rule)을 준수하는지 검증해주는 역할을 한다. 도구의 기능 범위는 JSLint, JSHint, ESLint 와 같은 기존 Open Source 기반 툴의 문법 검사 기능을 수용하고, 스마트 TV 플랫폼 특성에 따른 미지원 API, 확장 API 사용 및 웹 브라우저에서 지원되지 않는 기능에 대한 검증을 추가하였다. 스마트 TV 앱 개발자들은 이 도구를 사용하여 개발 비용이 적게 드는 초기 단계에 표준을 준수하는 앱을 구현 함으로서 TV 정합 시간을 줄일 수 있으며, 다른 디바이스나 플랫폼으로 앱을 이식하는 작업을 효율적, 경제적으로 할 수 있다.

1. 서론

인터넷이 고속화 되면서 데스크탑은 물론, 모바일, TV, 자동차 및 생활 가전 영역까지 웹에 접근하는 채널이 점차 다양해지고 있다.[1] 또한 모바일 디바이스가 빠른 속도로 성장하면서, 상이한 플랫폼과 OS 를 갖는 디바이스들 간의 높은 연결성과 호환성, 그리고 기기 간의 데이터 공유를 위한 처리 기술을 필요로 하는 미래형 융복합 서비스에 대한 요구가 증가하고 있다. 이에 따라, 이러한 요구사항을 실현하기 위한 앱 개발의 난이도와 복잡도도 함께 증가하고 있는 추세이다. 이런 환경에서 HTML5 는 크로스 플랫폼, 크로스 디바이스의 앱 개발을 가능하게 하여 한번의 개발로 여러 디바이스를 지원함으로서 개발의 난이도와 복잡도 문제를 해결하는 솔루션으로 급부상하고 있다.

이러한 웹 기반 서비스로의 환경 변화에 따라, 기존의 웹 기술과 웹 기술을 통해 하드웨어를 제어하는 디바이스 API 가 융화되도록 표준화가 진행되고 있으며, TV 또한 차세대 웹 플랫폼으로 HTML5 기술을 주목하고 있다. [2]

이러한 기술 동향과 관련하여, 한국 정보 통신 기술 협회(TTA)는 IPTV, DCATV, 지상파, 위성 등 다양한 방송 환경에서 HTML5 를 기반으로 제작된 스마트 TV 용 앱이 동작하기 위한 웹 브라우저 기반의 공통 플랫폼을 정의하는 “ HTML5 기반 스마트 TV 플랫폼” 표준[3] (이하, 스마트 TV 표준)을 제정하였다. 또한 이 표준을 기반으로 적합성 시험표준을 수행하기 위해, “HTML5 기반 스마트 TV 플랫폼 수신기 표준 적합성 시험” 표준[4]을 2013 년에 제정하였다.

본 논문은 스마트 TV 표준 기반의 TV 용 앱 생태계 활성화를 위한 시험 표준 개발 활동의 일환으로 앱 개발자 기술지원을 위해 제공하는 앱 검증도구에 관한 내용을 다룬다. 이 도구는 개발자의 통합개발환경(Integrated Development Environment, IDE)에 포함된 개발 툴의 하나로 제공되며, 구현된 앱이 표준에 맞는지 검증해 주는 역할을 한다. 앱 개발자들은 이 도구를 사용하여 개발 비용이 적게 드는 초기 단계에 표준을 준수하는 앱을 구현 함으로서 TV 정합 시간을 줄일 수 있으며, 다른 디바이스나 플랫폼으로 앱을 이식하여 정합하는 시간을 줄일 수 있다.

본 논문은 다음과 같이 구성된다. 2 장에서는 스마트 TV 용 앱 검증도구에서 지원해야 하는 검증 기능을 보고, 3 장에서는 스마트 TV 웹 앱 개발을 위한 검증도구의 개발 내용을, 4 장에서는 검증 앱의 실행 결과 내용을 다룬다. 마지막으로 5 장에서는 논문의 결론을 맺는다.

2. 스마트 TV 용 앱 검증도구의 문법도출

스마트 TV 의 브라우저에서 실행되는 HTML5 기반의 앱은 그 주요 구성 요소인 Javascript 특성과 스마트 TV 의 확장 API, 미지원 표준 API 등의 플랫폼 특성 때문에 검증도구가 반드시 필요하다. 이번 장에서는 그 필요성을 살펴보고 기존의 도구들이 필요한 기능들을 갖추고 있는지 확인하여, TTA 가 지원해야 하는 도구의 기능 범위를 살펴보도록 한다.

아래는 언어와 플랫폼 특성에 따른 검증도구의 필요성 및 도구에서 지원해야 할 기능이다.

[1] 본 연구는 미래창조과학부의 “방통융합 기반기술 테스트 환경구축” 과제의 일환으로 수행한 결과임

첫째, Javascript 는 컴파일 없이 동작하게 되므로 실행시간에 언어를 번역하는 엔진이 소스코드를 읽음과 동시에 동적으로 모든 사항을 처리한다. 그러므로, 실행 이전, 즉 구현시간에 소스 코드에 대한 기본 문법 오류를 검출해주는 도구가 필요하다. 둘째, Javascript 는 none typed 언어로 실행시간에 동적으로 자료 형을 검사하므로, 구현시간에 스마트 TV 앱에 포함된 TV 를 제어하는 확장(디바이스) API 의 자료 형이 제대로 사용되었는지 검증이 필요하다. 셋째, 스마트 TV 용 앱은 플랫폼 표준의 프로파일 정의에 따라 W3C 표준 API 중 일부를 지원하지 못하는 경우가 있다. 구현된 코드에 미지원 API 가 포함되었는지 검증이 필요하다. 넷째, Javascript 는 prototype 기반 언어로, 동적으로 객체에 자료와 메서드를 추가하거나 변경할 수 있다. 이 특징은 선언되지 않은 변수나 메서드가 사용되는 경우, 동적으로 메모리가 할당되어 undefined 로 초기화되어 사용된다는 것을 의미한다. 그러나 브라우저 내에서 사용되는 Javascript 엔진은 객체에 선언되지 않은 메서드를 호출하는 경우, 오류를 발생시켜 앱의 동작을 정지시킬 수 있다. 이러한 오류를 발생시킬 수 있는 코드가 있는지, 검증해야 한다.

다음은 기존의 open source 기반의 Javascript 검증 도구가 위 기능들을 포함하는지 비교 분석해본 결과이다.

기능 분류	JSLint[5]/ JSHint[6]	ESLint[7]	Web App Checker[8]
ECMAScript (기본문법검증)	5.1/6	5.1	5.1
브라우저의 Javascript 미 지원 속성 검증	미 지원	미 지원	미 지원
확장 API 검증	미 지원	미 지원	- Tizen API 검사 - 보안검사
규칙 변경/추가용이성	용이하지않음 (progressive)	용이함 (esprima)	용이하지 않음
미 지원 API 사용 여부 검증	미 지원	미 지원	미 지원

표 1. JSLint/JSHint, ESLint, Web App Checker 의 비교

JSLint, JSHint 와 ESLint 는 전형적인 Javascript 검증 도구이며, Web App Checker 는 삼성이 만든 플랫폼, Tizen 에서 제공하는 검증도구이다. 위 도표에서 알 수 있듯이, ESLint 는 JSLint/JSHint 에 비해 규칙 추가/변경이 용이하여, 새로운 기능들을 정의해야만 하는 디바이스 제조사에서 검증도구로 사용하기에 좋다. Progressive 와 Esprima 방식의 차이점은 전자는 소스 Parsing 과 규칙 처리를 한 모듈 내에서 순차적으로 처리하는 반면, 후자는 분리해 개별적으로 처리한다는 것이다. 분리해 처리하는 경우, 규칙 처리 부분만 수정하거나 추가하면 되므로, 기존 규칙에 대한 변경이나 새로운 규칙에 대한 추가가 용이하다. 그리고 Web App Checker 는 JSLint/JSHint/ESLint 와 다르게 Device API 를 검사하는 것이 특징이다.

이러한 비교 분석 결과를 바탕으로 TTA 에서 개발한 앱 검증도구의 기능 범위는 아래와 같다.

- ECMAScript 5.1 에서 정의한 규칙을 검증한다.
- 스마트 TV 디바이스를 위한 확장 API 사용법이 옳은지 검증한다.
- 브라우저가 지원하지 않는 Javascript 언어의 prototype 기반 언어 특성을 사용하는지 검증한다.
- W3C 표준에서는 정의되어 있으나, 스마트 TV 플랫폼 표준에서 지원하지 않는 기능을 사용하는지 검증한다.

3. 스마트 TV 웹 앱 개발을 위한 검증도구

가. 스마트 TV 앱 구동 환경

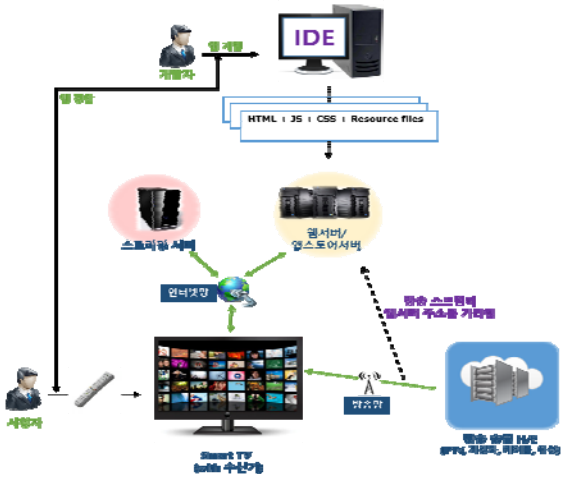

그림 1. Smart TV 앱 구동 환경

위 그림은 스마트 TV 플랫폼 표준에 따라 스마트 TV 앱의 구현부터 실행까지 구동 환경을 나타낸다. 스마트 TV 에서 앱이 실행되는 경로는 두 가지이다. 첫 번째는 사용자가 서버에 있는 앱을 TV 에 설치한 후 실행하거나, 두 번째는 TV 가 방송 스트림을 받아 스트림 내용을 화면에 표시할 때, 스트림 상에 앱의 위치가 들어 있어 그것을 가져와 실행하는 경우이다.

스마트 TV 앱은 개발자가 IDE 를 통해 구현하고, 정합 과정을 거친 후, 앱 서버에 저장한다. 구현 시 문제점을 찾지 못하면 정합과정을 통해 문제점을 찾아야 하는데, TV 에서 앱을 정합하며 문제를 찾는 것은 시간이 오래 걸리고 디버깅 또한 쉽지 않다. 그러므로, 개발자는 검증 도구를 사용하여 구현 시점에 문제점을 해결해야 한다.

나. 검증도구의 소프트웨어 아키텍처

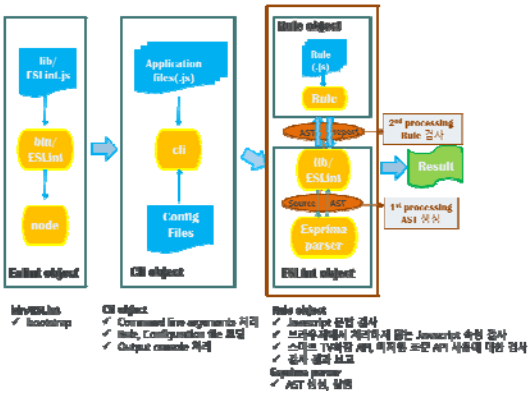

그림 2. 검증 도구 아키텍처

그림 2 는 TTA 에서 개발된 앱 검증도구의 아키텍처를 나타낸다. 네 개의 모듈로 구성되며, 각각의 기능은 그림 하단에 명시된 것과 같다.

검증도구는 ESLint 기반이며, 추가적으로 브라우저에서 처리하지 않는 Javascript 언어의 prototype 기반 언어 특성 사용 여부, 플랫폼 미지원 표준 API 사용 여부, 확장 API 사용법에 대한 검증을 추가하였다.

다. 검증 상세 Procedure

앱 검증도구의 검증 상세절차는 그림 3,4 에서 보는 것과 같다. 첫째로, 도구는 소스 전체를 읽어 토큰화 하고 토큰화한 결과를 구문해석 한다. 구문 해석은 프로그램 코드를 컴퓨터가 처리하기 쉬운 형태로 바꾸는 것을 의미한다. 둘째로, 도구는 구문 해석 결과에서 쓸데없는 토큰들을 없애는 의미해석을 한 후, 추상 구문 트리(AST, Abstract Syntax Tree)[9]를 생성한다.

그림 3. 1st Processing – 구문해석

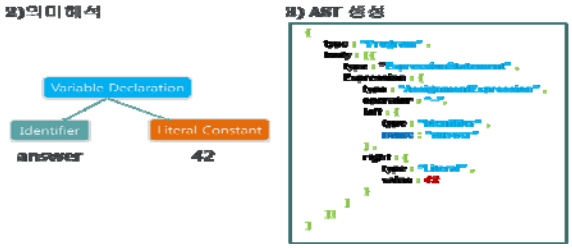

그림 4. 1st Processing – 의미해석과 AST 생성

셋째로, 도구는 그림 5 와 같이 생성한 추상 구문 트리(AST, Abstract Syntax Tree)를 Top Down 으로 읽어 가며 각 Rule 이 등록한 구문이 나온 경우, Rule 의 callback 함수를 호출해 준다. 각 callback 함수에서는 규칙을 검사하고 결과를 보고 한다.

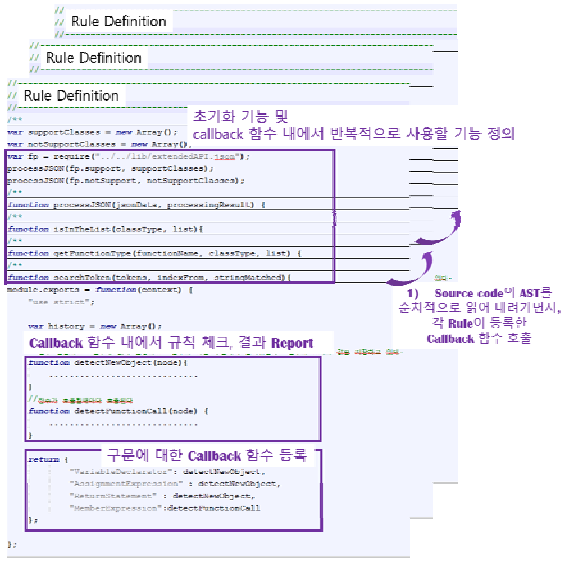

그림 5. Rule file 형식

그림 5 에서 보는 것 같이 Rule 파일은 callback 함수를 등록하는 부분과 규칙을 체크하여 결과를 보고하는 부분으로 구성된다.

도구의 Rule 처리 상세 Procedure 는 그림 6 의 순서도와 같다.

1. 플랫폼 표준 버전 별로 확장 API(테이블 1)와 지원하지 않는 API(테이블 2)가 다르므로, 이러한 API 에 대한 명세를 파일로 관리한다. 프로그램 초기에 이러한 파일의 명세를 읽어와 각각을 아래와 같은 테이블로 생성해 놓는다. Runtime 시 멤버가 추가되는 경우, 업데이트한다.

클래스	멤버	
객체 이름	멤버 변수	타입
	멤버 메소드	리턴 타입

표 2. API 의 Global member 정보 테이블

2. 변수 선언, Assign, 객체 표현, 함수 선언, 함수 표현 구문을 Callback 함수로 받아, 사용자 정의 API 를 만들고, 함수 호출 구문에 대한 Callback 함수를 받아 실시간 API 를 업데이트 함으로서, 그림 6 과 같은 테이블을 만든다.(테이블 3)

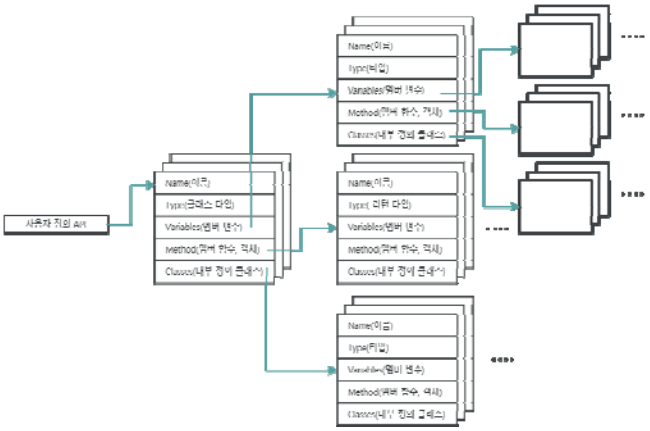

그림 6. 사용자 정의 API 테이블

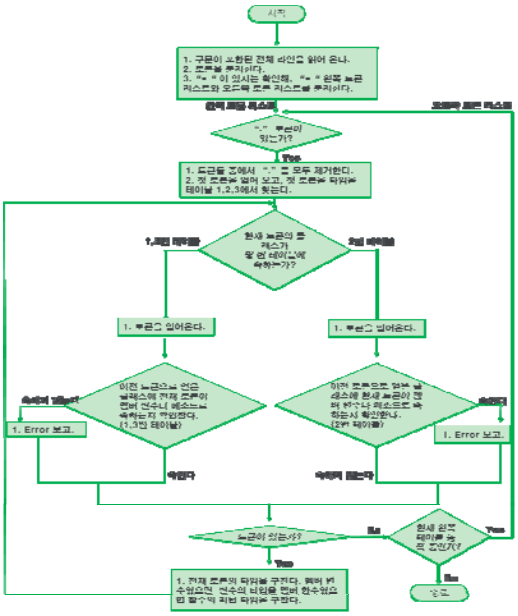

그림 7. 규칙 처리 시나리오

3. 멤버 변수, 멤버 메서드 호출 구문에 대한 Callback 함수를 등록한다. 이 함수가 호출되면, 그림 7 과 같이 멤버 데이타가 속한 클래스의 인스턴스가 생성되었는지

정의된 테이블을 통해 확인한다. 정의되어 있다면, 그 타입을 검사해, 그 타입이 확장 API 나 Runtime API 인 경우, 멤버 데이터가 맞는지 확인 후, 멤버 데이터가 아니라면 Error 를 표시해 주고, 그 타입이 플랫폼에서 지원하지 않는 API 인 경우, 멤버 데이터가 맞는지 확인 후, 멤버 데이터라면 Error 를 표시해 준다.

4. 도구 기능 테스트 및 결과

그림 8 은 앱 검증도구에서 스마트 TV 의 확장 API 와 미지원 API 에 대한 명세 파일이다.

그림 8. 플랫폼의 확장 API 와 미지원 API 에 대한 명세(.json)

확장 API 가 정의된 클래스 구조의 패스에 맞춰서 사용되는지 검증하기 위해, 각 API 를 사용하기 위해 얻어야 하는 모든 클래스에 대한 명세를 갖고 있어야 한다.

그림 9 은 개발자 테스트 코드 파일이다.

```
var context = document.getElementById('ttacanvas').getContext('2d');

context.moveTo(0,0);
context.lineTo(200,100);
context.stroke();

context.beginPath(); // right
context.arc(95,50,40,0,2*Math.PI);
context.stroke();

context.transform(1, 1, 2, 2, 3, 3); // wrong

var myTV = tvExt;
var myBroadcast = myTV.broadcast; //right
var myRadio = myTV.radio; //wrong

var user_obj = {};
user_obj.data = 9; //right
user_obj.test(); //right
```

그림 9. 테스트 코드 파일(.js)

스마트 TV 플랫폼에서 표준 2D canvas 상에 Context 클래스의 beginPath 메서드는 사용 가능한 함수이고, transform 함수는 플랫폼에서 지원하지 않은 메서드이다.

그림 10 은 테스트 결과 파일이다. 30 line 에서 transform 메소드가 플랫폼에서 지원되지 않는다는 메시지와 34 line 에서 tvExt.radio 가 확장 API 도 아니고 새롭게 정의된 API 도 아니라는 것과 38 line 에 Object.test 가 선언되지 않고 사용되었다는 것을 설명해준다.

그림 10. 테스트 결과

5. 결론

본 논문은 TV 용 앱 생태계 활성화를 위해 앱 개발자에게 제공하는 표준 및 검증 도구의 필요성을 알아보고, 검증해야 하는 규칙 도출, 소프트웨어 구조 및 규칙 구현까지 전반적인 스마트 TV 앱 검증 도구 개발 내용을 다루었다.

이 도구를 통해 앱 개발자들이 개발 초기 단계인 구현 단계에 웹 언어의 문법과 스마트 TV 플랫폼 표준을 준수하는 앱을 구현 함으로서 앱의 TV 정합 시간을 줄이고, 다른 디바이스나 플랫폼으로 앱을 이식할 때, 정합하는 시간을 줄일 수 있길 기대한다. 또한 이 논문이 HTML5 기반 디바이스 플랫폼 개발사에게 새로운 Rule 을 쉽고 빠르게 개발할 수 있는데 참조가 되는 기초 문서로서 활용되길 바란다.

현재 도구는 “ Javascript 언어의 prototype 기반 언어 특성” 에 대한 기능 검증을 부분적으로만 지원하고 있다. 앞으로 TTA 에서는 이 기능을 전체적으로 지원하도록 업그레이드 할 계획이다.

참조문헌

[1] Appcelerator, IDC, 2015 년 사용 예상 기기, 2013

[2] 한국 방송 통신 전파 진흥원, HTML5 기반 스마트 TV 플랫폼 산업 동향, 2013.7.3

[3] TTAK.KO.07-0111/R1, “HTML5 기반 스마트 TV 플랫폼”, 2014.04

[4] TTAK.KO.07-0119, “HTML5 기반 스마트 TV 플랫폼 수신기 표준 적합성 시험”, 2013.12

[5] JSLint Homepage, http://www.jslint.com,

[6] JSHint Homepage, http://www.jshint.com/

[7] ESLint Homepage, http://eslint.org/

[8] Tizen Homepage, https://source.tizen.org/ko/compliance/application-compatibility/web-application-checker?langredirect=1

[9] MDN mozilla developer network Homepage, https://developer.mozilla.org/en/SpiderMonkey/Parser_API