Open Source기반 HTML5 Mobile Web Application Platform 구조 분석 및 성능 최적화 방법

임상석
SK 플래닛

요 약

본고는 크게 두가지 주제로 구성이된다. 첫번째로는 HTML5 기반의 mobile Web application platform 구조에 대해서 상세히 소개한다. Web application platform은 기술 구조상 mobile OS에 내재되어 native형태로 배포되는 Browser engine을 포함한 platform 부분과 native Web platform 상에서 구동되는 HTML5 application framework 부분으로 구성된다. HTML5 application framework 구축을 위해 시장에서 널리쓰이는 open source로서 jQuery Mobile framework을 소개한다. 두번째로 해당 Web platform상에서 동작하는 Web application 개발시 부디칠 각종 성능 이슈 및 그것을 해결하기 위한 접근법을 다섯가지 기술 영역으로 나누어, 각 영역별로 적용 가능한 실기를 다룬다. 마지막으로 최적화시 사용가능한 각종 open source profiling 및 성능 최적화 tool에 대해서 소개한다.

Ⅰ. 서 론

Mobile 환경에서의 HTML5의 급속한 확산은 수많은 조사에[1], [2], [3] 의해서 예측되어지고 있고 현재 진행중이다. 최근 조사로서 ABI Research[1]에 따르면 2016년까지 21억대의 mobile 단말이 HTML5를 지원하고 현재 iOS와 Android의 native mobile OS의 smart phone platform로 양분된 경쟁구도에 대해 cross-platform을 핵심 차별점으로 하여 큰 변화를 주도할 것으로 예측하고있다.

HTML5는 Web의 content 를 구조화하고 표현하는 markup 언어이지만[4], Web platform 기술 분야에서는 단순히 하나의 표준외에 HTML5와 결합하여 다양한 Web application을 개발시 필요로하는 SW platform으로서의 모든 관련 기술 표준 명세서[5]를 일반적으로 통칭한다. 즉, HTML5기반 Web platform이란 markup으로서의 HTML5, 콘텐츠 표현 방식의 확장인 CSS3, 그리고 각종 application 구현을 위한 audio, video, sensor, GPS등과 같은 기능 요소 접근을 위한 JavaScript API들로 구성이된다.

HTML5의 확산은 smart phone용 mobile OS에서 해당 표준을 지원하는 browser엔진의 도입과 병행이되어야만 실질적인 상용화 적용이 가능하다. 최근 각광을 받고 있는 smart phone OS인 iOS, Android, Blackberry OS, 바다, Tizen에서 채택한 browser layout엔진인 WebKit[6]은 Internet Explorer에 비하여 매우 높은 호환성을 제공하고 WebKit의 mobile OS의 탑재율이 90%를 초과하여 호환성이 낮은 Internet Explorer가 주류인 desktop환경보다는 mobile에서의 지원도가 매우 높아 HTML5는 desktop 환경보다는 mobile에서 확산이 더 빠르게 진행되고 있다. 이에 다수의 mobile용 HTML 5 framework이 open source로 활발히 개발되고 있다[7][8][9]. 본고에서는 이중 가장 활발한 개발자 ecosystem을 확보하고, 가장 넓은 단말를 지원하는 jQuery Mobile을 UI framework로서 상세히 소개한다. Desktop환경도 최근 WebKit을 기반으로하는 Google사의 Chrome browser가 시장 점유율 1위를 기록하고, Internet Explorer 10부터는 높은 HTML5 호환성을 지원함으로 조만간 확산의 기반이 다져질 것으로 예상된다.

jQuery Mobile을 포함한 모든 종류의 HTML5기반 UI framework을 활용하여 개발된 mobile Web application은 그것의 구동 및 배포 환경에 따라 크게 두가지 형태로 〈그림 1〉과 같이 나눌수 있다.

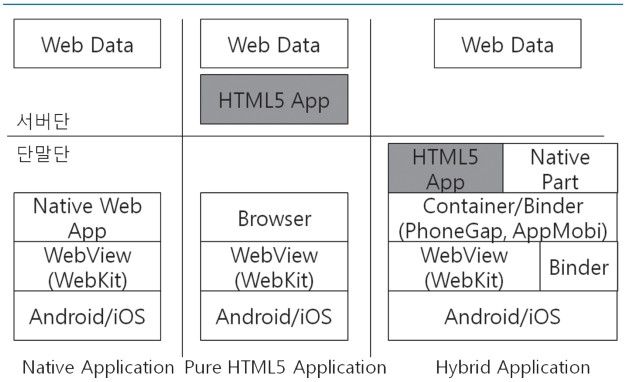

그림 1. Web application 배포 방식별 분류

첫번째로는 기존의 Web site의 형태로 UI를 포함한 모든 application 요소가 서버에 위치하고 URL을 통하여browser로 download되어 구동되는 형태이고, 두번째로는 hybrid Web application으로 불리우는 방식으로 일반 native application 개발방식과 동일 방식으로 하나의 application package로 만들어지고, 해당 package내에 HTML, CSS, JavaScript구현부와 native code구현부가 통합되어 포함된 형태로서 application store를 통하여 배포되고 smart phone의 저장공간에 설치되는 방식이다. 두번째 방식에서는 application 구동을 위하여 browser가 활용되지 않고 WebView (WebKit 엔진을 widget화하여 3rd party API로 제공하는 component)를 활용하여 설치 package에 포함된 HTML, CSS 및 JavaScript를 적재하고 처리한다. 본고에서 소개하는 기술은 기본적으로 첫번째와 두번째 방식 모두에 적용 가능하다.

산업 현장에서 HTML5기반 application또는 서비스를 상품화시 native application 대비 경쟁력 있는 SW 품질을 달성하기 위해서는 위 두가지 방식으로 개발된 mobile Web application의 성능 최적화는 단말 부분과 서버와 연동 부분을 모두 아우르는 end-to-end 최적화가 필수적이다. End-to-end를 고려하지 않는 단편적인 성능 개선은 실질적인 application 성능 개선효과가 반감이 되기 때문이다. 예로써, Rendering 성능 관점에서 최적화된 HTML, CSS, JavaScript를 구현하였더라도, network을 통한 배포관점에서 이것들이 변경사항이 없음에도 불구하고 단말에 cache되지 않거나 내재된 파일들이 병렬로 동시에 network을 통한 download 되지 않는다면 전체 성능 관점에서는 매우 떨어지게 된다. 이에 본고에서는 end-to-end 성능에 영향을 주는 각종 지연 요소를 분류하고 지연 요소별 연관된 근본 기술 원인을 소개한다. 이를 근간으로 최적화 방법론에 대해서 practice 위주로 정리하였다. 그리고 추가하여 이러한 최적화시 활용 가능한 open source profiling tool들을 정리하였다.

전체 구성은 다음과 같다. II장에서는 HTML5 mobile Web application platform architecture중 mobile OS에서 제공되는 native부분에 대해서 상세히 설명하고, III장에서는 이것 상에서 구축된 HTML5 application framework 부분에 대해서 소개한다. IV장에서는 end-to-end 최적화 방법에 대해서 practice 위주로 설명하고 V장에서 결론을 맺도록 하겠다.

Ⅱ. HTML5 Mobile Web Application Platform: Native mobile OS부분

HTML5 Web application platform은 〈그림 2〉에서와 같이

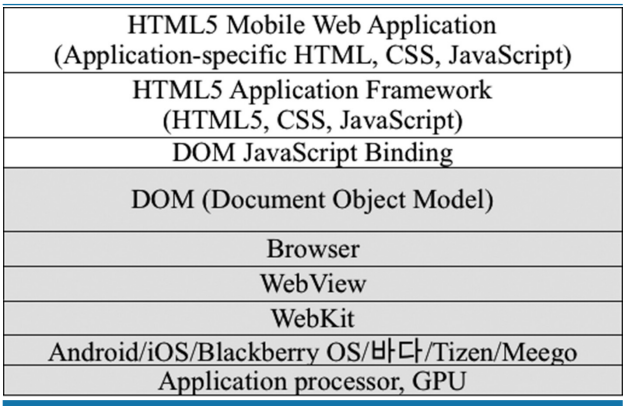

그림 2 개념적 Web platform SW 계층도

다계층 구조로 이루어지고, 다시 구현 및 배포방식에 따라 크게 두가지 계층으로 구분된다.

회색 음영 처리된 부분으로 단말의 OS종류에 따라 구성되는 Native 부분과 그상에서 구동되면서 실제 HTML, CSS, JavaScript로 HTML5 application 작성시 필요한 HTML5 application framework 부분이다. Native platform 부분은 platform 업체 또는 단말 제조 업체가 HW 단말에 탑재하여 배포하는 부분으로 구현 변경이 사실상 불가능한 부분이나 실제 HTML5 application framework을 구축시 HTML5 application의 runtime을 제공하고 각종 성능 최적화시 필요한 기술 범위에 포함되는 부분으로서 깊은 이해가 필수적이다.

〈그림 2〉에서와 같이 이 두 계층은 WebKit엔진에서 제공하는 DOM의 JavaScript binding layer를 중심으로 나뉜다. Programming관점에서 Web application은 Web platform에서 제공하는 모든 서비스를 DOM JavaScript binding을 통해서 접근한다. HTML5 application framework은 실제 HTML 5 표준에서 제공하는 각종 HTML5 markup, CSS3 그리고 JavaScript API로 binding 되는 부분과, 표준 이외에 application작성시 필요한 JavaScript library들로 구성 된다.

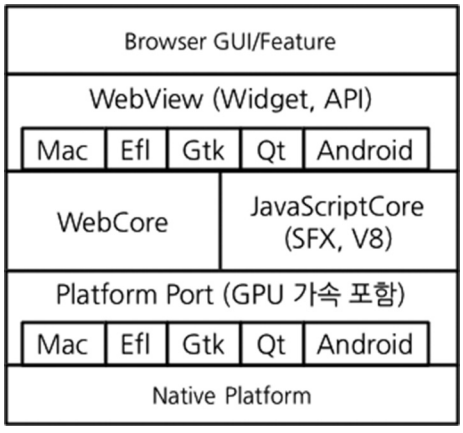

그림 3. WebKit 기반 native 부분 상세도

첫번째로 Web application platform의 native를 담당하는 부분을 아래 〈그림 3〉에 좀더 구체적으로 도시하였다. WebKit 엔진은 HTML, CSS, JavaScript의 parsing, rendering, execution을 담당한다.

Native Web platform은 크게 OS별로 Widget 및 API를 제공하는 WebView와 HTML, CSS, JavaScript를 parsing하고 실행시키는 WebCore와 JavaScript엔진, 그리고 parsing된 결과를 화면에 실제 rendering하는 platform port로 구성이된다. WebKit자체는 open source로서 상당 부분의 코드를 공개하여 공유 하지만 〈그림 3〉에서의 platform 이름 별로 각각의 platform 별도의 구현부가 존재한다. Platform 별 실질적인 browser 및 Web application의 성능 차이는 platform port부분의 제조사별 최적화 정도에 따라 주로 발생한다. Apple사의 Mac OS와 iOS용 port는 open source로 공개되어 있지 않다. Native platform 구성요소중 IV장에서 성능 최적화를 설명하기 위해서 실제 Web application 개발시 programming 방식에 따라 성능 차이가 발생하는 WebKit의 주요 모듈에 대해서 다음 절에서 상세히 설명한다.

본절에서는 HTML5 Web application 개발시 성능과 밀접한 관련이 있는 내부구조에 대해서 살펴본다. WebKit엔진은 입력받은 HTML과 CSS를 parsing하여 중요한 두개의 data structure인 DOM tree와 Render tree를 생성한다. DOM tree[10]는 HTML document를 접근 및 변경 할 수 있도록 표준 규격에 따라서 JavaScript object 형태로 Web application에 제공된다. 이러한 DOM tree는 HTML이 기술한 document의 구조를 유지하고 WebKit은 DOM tree와는 별도로 실제 화면에 rendering시 사용하기 위하여 Render tree를 구성한다. 이는 HTML parser에서 DOM tree에 새로운 node를 attach할 때마다 RenderObject를 생성하여 Render tree에 삽입하여 구성한다. DOM tree의 각 object node중 화면에 공간을 차지하고 실제 rendering이 되는 node들만이 상응하는 Render tree의 Object를 생성 시킨다. CSS 속성중 display:none통해서 화면의 공간을 차지하지 않게 설정하면 해당 DOM node는 계속 유지하지만, 해당 RenderObject는 Render tree에서 제거되게 된다. 이러한 동작 방식은 추후 Scroll성능 최적화시 꼭 알아야 할 기본 지식이다. Scroll과 같은 연산에 의해서 화면을 갱신할 때는 DOM tree나 Render tree의 구조의 복잡도 또는 node 수가 많을 수록 소요시간이 증가하게 된다.

```
〈html〉
    〈head〉
    〈title〉SK Planet〈/title〉
    〈style〉…〈/style〉
    〈script〉…〈/script〉
    〈/head〉
    〈body〉
        〈div〉
            〈span〉
                〈a href="abc.com"〉News〈/a〉
            〈/span〉
            〈span〉…〈/span〉
            〈/div〉
     〈/body〉
〈/html〉
```

위표의 HTML markup을 입력으로 하여 WebKit엔진에 의해서 생성된 DOM tree와 Render tree를 〈그림 4〉에 도시하였다. DOM tree의 각 node중에 화면에 실제 표시되는 부분에 해당하는 "News"와 그것의 부모 node들만으로 구성된 subtree형태를 구성하는 RenderObject들로 전체Render tree가 구성되어 있음을 확인 할수 있다. 참고로 DOM에서 제공하는 Event Capturing과 Bubbling은 DOM tree만을 traverse[10]하여 수행한다.

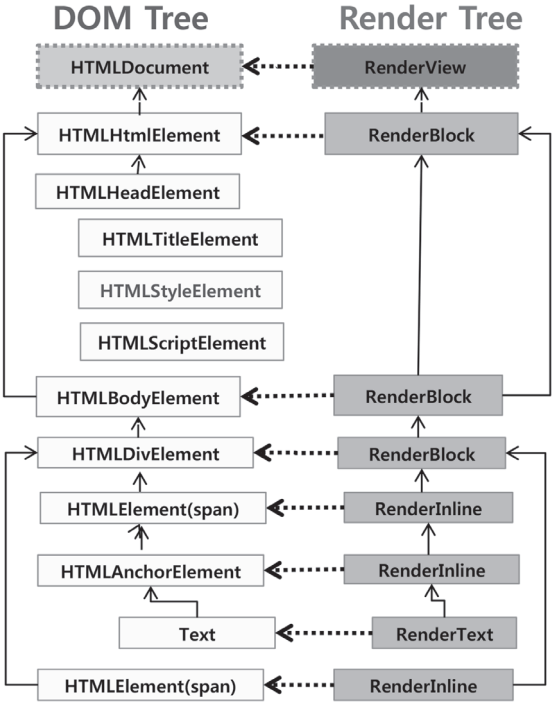

그림 4. DOM 및 Render tree 생성 예제

위와 같이 Render tree가 구성되면 WebKit엔진은 구성된 Render tree를 traversing하며 RenderObject의 내용에 따라

화면에 실제 rendering을 수행하게 된다. 이러한 Render tree의 생성후에는 사용자의 action또는 JavaScript등에 의해서 화면 갱신을 수반하는 연산 결과를 필요에 따라 rendering을 수행한다. 이러한 rendering을 위하여 WebKit엔진은 Render Layer tree라는 것을 추가적으로 생성한다. RenderLayer는 일반적으로 화면에 그릴때 그리기의 속성이 유사하여 한번에 같이 그려낼때 효율적인 Render tree의 subtree들로 분리하여 다시 tree로 구성된다. Render tree에서 RenderLayer tree를 구성시 개별 RenderLayer의 생성 조건은 아래와 같다.

- 전체 page에 대한 root (최소 한개 기본 생성)
- 명시적으로 CSS position property 설정 (relative, absolute, transform)
- Transparency 값 설정
- overflow, alpha mask, reflection 설정
- 〈canvas〉 element
- 〈video〉 element

최근에 WebKit에 구현된 기술인 HW Accelerated Compositing (AC)은 위와 같이 분리 된 RenderLayer를 기본 단위로 합성을 수행한다. HW AC는 일반적으로 OpenGL ES를 지원하는 GPU(Graphics Processing Unit)에서 RenderLayer

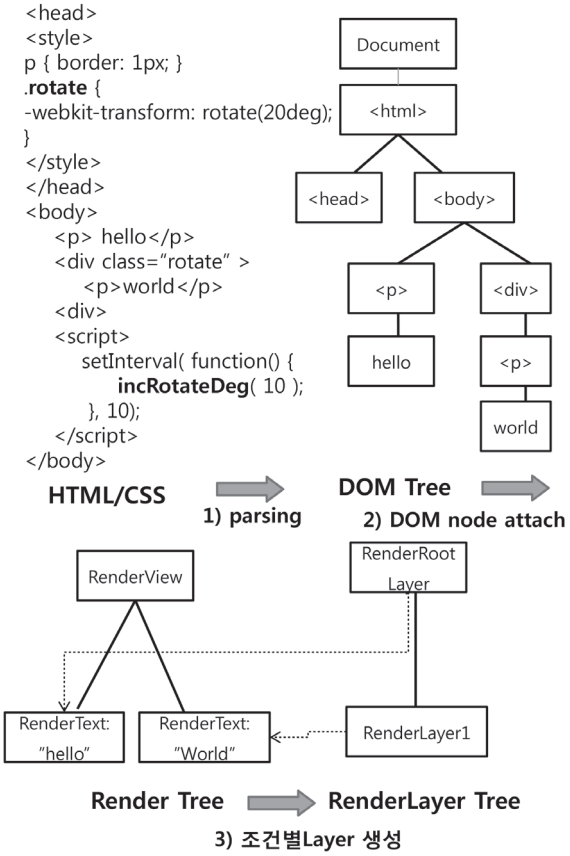

그림 5. RenderLayer Tree 생성 예제

간 고속 compositing을 수행하는 기술이다[19]. HW AC가 지원되는 WebKit에서는 개별 Render Layer별로 GPU에 의해서 접근되는 texture 메모리를 별도로 생성하여 mapping하고, 해당 texture 메모리에 CPU로 해당 RenderLayer에 포함된 content를 출력하여 저장한다. 이렇게 생성된texture는 추후에 compositing 연산시 활용이 되며 해당 RenderLayer의 content가 invalidate되거나, 메모리 부족으로 texture가 flush되지 않는한 추후 rendering시 해당 RenderLayer의 content를 CPU를 통해서다시 rendering하지 않고 계속 재활용이 된다.

〈그림 5〉에서는 설명한 내용의 이해를 돕기 위하여 Render Layer가 생성되는 절차를 예제를 통하여 도시하였다. Root Layer한장은 기본 생성이 되고 rotate class에 적용된 -webkit-transform 속성에 의하여 "world"라는 content를 갖는 RenderLayer가 추가로 생성이 되어 Tree에 포함되어 크게 두개의 RenderLayer로 구성된다. 이렇게 생성된 RenderLayer tree를 활용하여 실제 화면에 Rendering하는 방식은 〈그림 6〉에 나타냈다. 먼저 좌측 방식을 살펴보면 HW AC를 지원하지 않는 WebKit엔진에서는 CPU가 한장의 canvas (rendering 결과를 저장하는 메모리로 개념적으로 framebuffer로 출력되는 버퍼)를 생성해서 Root RenderLayer와 RenderLayer1을 통해서 최종 출력용 이미지 한장을 즉시 만들어 낸다. HW AC가 지원되는 WebKit에서는 각 Render Layer별로 총 두장의canvas를 생성하여 각각에 대해서 별도의 canvas로 rendering을 수행한다. 이때 "world"를 포함한 canvas는 GPU의 texture로 생성이 된다. 이렇게 생성된 두장의 canvas는 GPU를 사용하는 HW compositor를 통하여 화면에 출력할 최종 image한장을 생성한다.

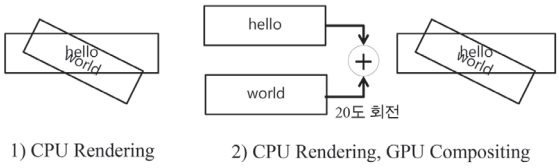

그림 6. HW Accelerated Compositing 동작

이때 성능에 가장 결정적인 영향을 주는 상황은 〈그림 5〉의 좌측 상단의 HTML예제에서 구현된 〈script〉 block을 통하여 animation효과를 줄때 명확히 설명 가능하다. 해당 예제는10msec주기로 incRotateDeg(10)을 호출하여 "world"를 포함한 canvas를 10도씩 추가로 회전시켜 animation을 구성하고자 할때 HW AC를 지원하지 않는 방식은 "Hello", "world" 두 부분을 전체를 매번 Render tree를 traversing하여 다시

rendering하여 최종 image를 만들어내는 반면에 HW AC를 지원하는 WebKit에서는 별도의 rendering은 발생하지 않고 단순히 "world"를 포함하는 texture를 GPU를 통하여 회전시킨이후 compositor로 합성만 해서 최종 image를 만들어낸다. HW AC를 지원하는 방식은 이러한 animation에 대해 HW AC를 지원하지 않는 버전에 비해 수배까지 빠른 성능을 보여준다.

지금까지 설명한 내부 동작 방식을 잘 이해하고, mobile Web application을 개발할때는 주어진 WebKit의 HW AC의 지원여부를 판별하고 이에 따라 GPU를 최대한 사용하는 방향으로 code를 구현하여야 충분히 빠른 성능을 달성할수 있다. 그러나 HW AC 방식은 지정된 Render Layer를 texture라는 추가 메모리를 사용하여 저장하므로 메모리 사용량이 늘어나는 단점이 있다. Mobile OS별로 메모리 사용량 개선을 위해서 Texture Atalas 기법을 도입하기도 하므로 단말별 상세 분석이 필요하다.

본장에서는 Web platform의 native단을 구성하는 요소 중 Web application의 성능과 밀접한 연관이 있는 기술에 대해서 살펴 보았다. Web application개발자가 변경할 수 없는 부분이지만 그것의 동작 특성을 충분히 이해하고 code를 개발해야만 성능 좋은 application 을 개발할수 있음을 이해하였을 것이다. 다음장에서는 이러한 native platform 상에서 동작하는 HTML5 application framework 구조에 대해서 설명하겠다.

III. HTML5 Mobile Web Application Platform:HTML5 application framework

Smart phone 유통시 이미 내장된native Web platform 부분과는 다르게 HTML5 application framework은 W3C에 의해서 표준화된 API와 application을 개발시 필요한 비표준 library들로 자유롭게 구성이 된다. 비표준 library는 설계에 대한 자유도가 매우 높은 부분으로 현재 다양한 형태의 SW 구조가 정의되고 구성될 수 있으나, 본고에서는 가장 널리 쓰이는 open source들을 활용하여 그것의 구조를 살펴 본다. 표준 API는 WebKit에서 구현되어서 DOM을 통해서 노출되므로 변경 불가능한 부분이다.

HTML5 application framework의 기반 기술은 HTML mark-up, CSS3, JavaScript로 구성이 되고 HTML5 표준에서 정의한 모든 API는 JavaScript를 통하여 접근이 가능하다. 〈그림 7〉에는 현재 W3C에서 주도하여 HTML5 application 개발시 필요한 각종 API 표준화 작업을 기능별로 분류하여 나열 하였다[11]. 현재 대부분은 표준화 작업이 진행되고 있어서 그것을 바로 적용하려면 실제 browser별로 지원여부를 필수적으로 파악하여야 한다. 〈그림 7〉에 나열된 표준의 상당수는 현재 시점에서 제공되지 않고 있으나 HTML5의 표준화가 완료가 예상되는 추후 2-3년 내에는 모두 사용 가능해질 것으로 예상된다.

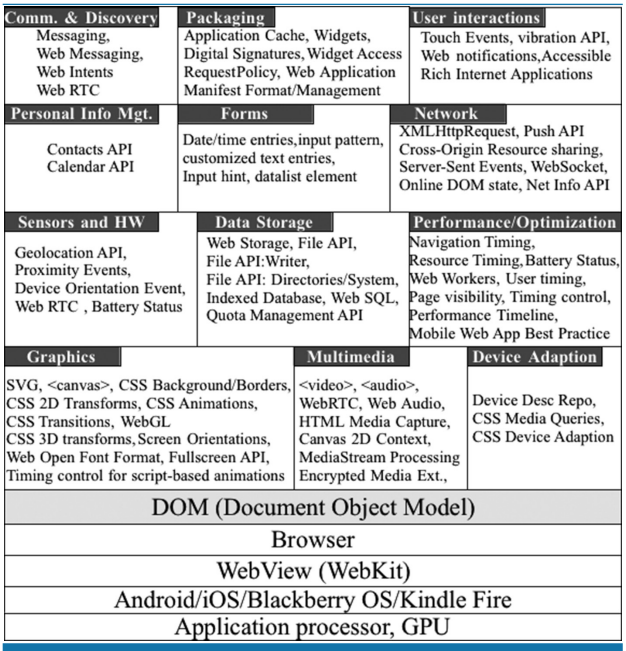

그림 7. Web application용 표준 Stack

Application 개발시 필요한 비표준 library로서 〈그림 7〉과 같은 표준API이외에도 UI를 구성 및 각종 event를 처리하는 UI framework이 필요하고 DOM 접근을 abstraction해주는 JavaScript library도 필요하다. 그러므로 application의 개발 관점에서 필요한 비표준 주요 요소로서 아래의 사항들을 추가로 분석하도록 하겠다.

- DOM 접근 library
- HTML5 UI framework

먼저 DOM 접근용 library로 가장 널리 쓰이는 jQuery[12]는 CSS selector를 편리하게 사용가능하게 하고 method chaining기법을 통하여 생산성을 향상 시킨다. 현재 대다수의 HTML5 application 및 Web site 제작에 사용된다. 단점으로는 기본적으로 DOM API의 직접 접근에 비하여 예제에 따라 최대10배가량 속도가 느리다. 그러나 활성화된 community, 재활용이 용이한 구조등으로 대부분에 web site 및 application 개발에 널리 사용된다. 속도가 매우 민감한 경우에는 DOM을 직접 접근하는 방식을 사용해야 한다.

두번째로 UI framework으로 널리쓰이는 open source로는 jQuery Mobile[7]과 Sencha Touch[8]가 있다. 두가지 방식의 기술적인 측면에서 가장 큰 차이는 jQuery Mobile은

markup 기반이고, Sencha Touch는 Ext.JS라는 JavaScript component를 기반으로 코드를 작성한다는 것이다. Mark-up 기반은 기존의 Web site를 개발하는 방식인 publisher가 styling된 HTML을 제공하고 programming logic를 JavaScript 개발자가 추가 개발하는 방식에 적합하다. Ext.JS 기반은 모든 UI component가 JavaScript component를 활용해야하므로 기존 publishing 방식의 프로젝트에 적용이 난해하고, publisher와 JavaScript 개발자가 같이 개발하거나, JavaScript 개발자가 주도적으로 개발하는 등의 방식으로 진행을 하여야 한다. 본고에서는 license자유도가 높고 좀더 개방적이며, Web publisher친화적인 jQuery Mobile에 대해서 좀더 살펴 보도록 하겠다.

	jQuery Mobile	Sench Touch
개발사	Jquery Open source community	ExtJS 개발사
개발자 Pool	125 개발자, 9 회사 sponsor	11 Sencha 직원 개발자
개발 방식	Mark-up기반 (Web Designer 친화적)	Script 기반 (JS programmer 친화적
지원 Platform	iOS 3.2+, Android 2.x+ Window Phone 7.x, Blackberry 6.x+, Palm Web OS 1.4+, 바다, Meego, symbian	iOS 3.0+, Android 2.1+, Blackberry 6.0+
개발 난이도	중	상
테마특징	Theme roller	Sass 기반
Library 크기	JS(80KB), CSS (48KB)	JS (92KB,) CSS (143KB)
License	MIT/GPL 선택	Commercial S/W
		License (무료) Commercial OEM License (유료)

UI framework의 여러 기술 요소중 가장 기본적인 검토 사항으로는 기존의 mobile Web page의 개념에서 mobile Web application으로 진화에 따라 touch에 최적화된 UI component를 제공하는지와 application 개발시 단일 HTML에 다수의 view를 구성 가능하고 view간의 전환 지원 여부이다. 이 두가지 사항에 대해 각각 살펴보겠다.

jQuery Mobile에서 제공하는 UI component는 크게 Dialog, Toolbar, Button, Form, List view를 대분류를 제공하고 있다. 각 분류는 세부 component를 제공하고, 전반적으로 native application의 GUI를 개발하기 위한 기본 component는 모두 제공하고 있으며, 이에 더하여 기본 theme 및 theme roller[13]를 통한 custom theme도 제공한다.

일반적인 Web page는 page별로 별도의 HTML 파일을 network으로부터 loading 한다. Session내에서의 data 공유는 cookie나 local storage를 활용하는데, 이러한 기법으로 Web application을 구현시 page전환시 network loading지연 발생으로 인한page 전환 지연과 page 전환시 history관리, 그리고 page간 data 공유의 어려움으로 application을 개발자의 개발 생산성을 제약한다. 이러한 점을 극복하기 위해서application의 view라는 개념이 HTML에도 도입이 되어야 하고 jQuery Mobile에서는 page라는 개념으로 제공된다. 기본적으로 하나의 application을 구성하는 모든 page는 하나의 DOM tree로 구성이 되어야만 page간 전환시 속도 문제가 없고, page간 application data 공유가 용이하다.

〈그림 8〉에서와 같이 하나의 HTML내에 application을 위한 다수의 view가 구성이되고 이는 하나의 DOM tree로 만들어진다. jQuery Mobile에서는 AJAX loading이라는 방식을 제공해서 별도의 HTML에 저장된 page를 동적으로 읽어서 DOM injection하는 방식 또한 제공한다. 이러한 방식은 접근하지 않는 page는 DOM tree로의 lazy loading을 통하여 메모리를 효율적으로 사용가능 하게하고, 각 page를 별개의 HTML파일로 저장 가능하도록하여 program의 유지보수성을 높일수 있다.

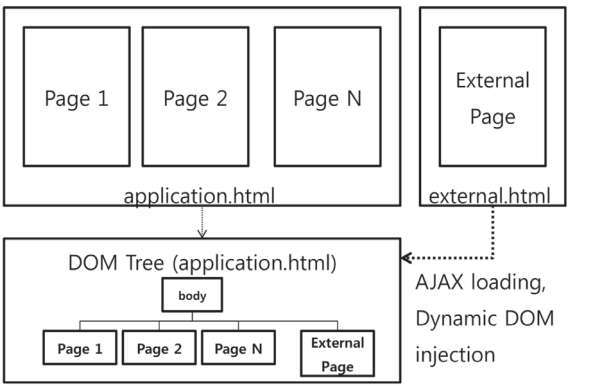

그림 8. 단일 HTML에서 복수의 Page지원방식

jQuery Mobile에서 page 구현 방식 및 활용법에 대해서 좀더 살펴보도록 하겠다. jQuery Mobile에서 page의 개념은 〈div〉라는 logical block과 HTML5 확장 tag인 data- attribute를 이용해서 구현 가능하며 아래의 예제에서 확인 가능하다. "foo"와 "bar"라는 id를 갖는 두개의 page를 하나의 html 파일에 생성한 예제로 data-role="page" 선언을 통해서 단일 page로의 역할을 하게 되고 이를 위해 기본 page처리 관련 event등을 내부적으로 자동 등록하게 한다. 이때 두 page간의 전환은 〈a〉tag를 확장하여 click시 data-transition에 명시된 transition 효과를 보여주면서 수행한다.

```
〈body〉
〈!— Start of first page —〉
〈div data-role="page" id="foo"〉
    〈div data-role="content"〉
    〈a href="#bar" data-transition="pop"〉 to bar〈/a〉
    〈/div〉
〈/div〉 〈!— page —〉
〈!— Start of second page —〉
〈div data-role="page" id="bar"〉
    〈div data-role="content"〉
    〈a href="#foo" data-transition= "slide"〉 to foo〈/a〉
    〈/div〉〈!— content —〉
〈/div〉〈!— page —〉
〈/body〉
```

jQuery Mobile은 다양한 UI component를 제공하고, page라는 개념 제공하여 쉽게mobile Web application을 개발 가능하도록하는 open source HTML5 framework이다. 다양한 custom UI component의 개발은 jQuery UI plug-in 형태로 개발 가능하다[18].

본장에서는 open source인 jQuery Mobile을 활용하여 UI를 개발하는 방식에 대해서 설명하였다. 다음장에서는 개발된 Web application의 성능 최적화 기법에 대해서 설명하겠다.

IV. End-To-End Web application 성능 최적화

Mobile Web application 구동시 서버에서 단말단까지 구성되는 지연 요소는 크게 단말에서의 HTML,CSS,JavaScript처리와 서버에서의 처리 지연 그리고 network 전송 부분으로 구성되고 상세하게 W3C에서 개념화하여 도시하였다[18]. 본장에서는 위 세가지 부분을 아우르는 성능 최적화 방법에 대해서 소개하고, 이를 위하여 사용 가능한 open source tool을 소개한다.

산업 현장에서의 성능 최적화 방법은 platform 별로 다양한 practice와 경험들의 집합으로 구성이 된다. 저자의 경험에 비추어 기반 기술 관점에서는 그것들을 아래와 같은 방식으로 크게 구분이 가능하다.

Parsing
1) Parallel resource loading 극대화 • document.write() 미사용 • iframe, CSS @import 미사용 • external CSS는 external JS 이전에 적재 2) Parsing 비용 절감 • HTML/CSS/JavaScript minifying
Layout 및 Rendering
1) Layout 발생 빈도 및 비용 최소화 • DOM tree node 개수 최소화 • DOM, CSS display property, CSS box model값의 변경 최소화 2) CPU 과도 사용 기능 사용 최소화 • Rounded corner, shadow, background-repeat, gradient 등
Graphics/Animation
1) HW AC 지원 platform에서 CSS 2D/3D animation 효율적 사용 • Animation 대상 Layer개수 및 크기 최소화 • Animation Layer의 content 내용 미변경 2) HW AC 미지원 platform에서 animation 사용 최소화
JavaScript
1) JavaScript loading timing 최적화 • 첫 화면 구성시 즉시 필요한 JavaScript와 필요하지 않은 JavaScript를 분리하여 lazy loading 2) JavaScript 수행 코드 수행 시간 짧게 유지 • JavaScript 수행동안 main UI thread는 중지됨 3) DOM manipulation 방식 효율화 • DOM 연산 결과 caching • DOM 변경시 개별 node단위 반복 변경 보다는 batching 작업 • DOM node 변경은 offline에서 수행 (변경node detach → 연산 → attach)
Networking
1) Network request 개수 줄이기 • CSS sprite 사용 • DATA URI로 문서에 resource 포함 • Local storage 및 HTTP cache 적극 활용 • CSS, JavaScript 파일 통합 2) Network RTT 최소화

위 표에서는 분류한 분야와 가장 중요한 guideline 그리고 해당 guideline을 수행하기 위한 practice들을 나열하였다. Browser engine 기술이 진화함에 따라 practice 또한 계속 바뀔 것이나 현 시점에서는 유효한 접근법들이다.

위 표와 같은 practice를 적용하기 위해서는 실제 작성된 application을 profiling 하고, 최적화 작업을 도와주는 tool들의 사용은 필수적이다. 다음 표에서는 바로 활용가능한 open source tool들에 대해서 특징 및 활용 방식에 대해서 간략히 정리하였다.

Google Page Speed [15]
Google에서 정한 best practice를 기준으로 작성된 application 적용 여부를 100점 만점으로 제공하고 실제 적용 guideline을 제공하여 매우 유용함
Google Speed Tracer[21]
Parsing, execution, layout,CSS style recalculation, selector matching, DOM event등 상세 동작정보를 visualization
Closure Compiler [22]
JavaScript code를 분석후 minify, dead docde 제거등 최적화 수행 Whitespace only, simple optimization, advanced optimization의 3단계 최적화 수준 선택 가능
DOM Monster [16]
Bookmarklet 형태로 제공되는 DOM 구성 내용 상세 분석 및 출력 불필요한 DOM 개수를 출력해줌

CSS Sprite Generator [23]
여러장의 image를 한장으로 묶어주는 기능
Chrome Developer Tools [17]
DOM inspection 기능 timeline으로 parsing, style calculation, garbage collector, JavaScript 수행 정보를 출력 WebKit 내부에서 CPU usage profiling 기능

본장에서는 HTML5 Web application의 성능 개선을 위해서 살펴보아야할 기술 분야와 해당 기술 분야별 성능 최적화를 위한 practice와 이때 필요한 profiling 및 최적화 tool에 대해서 간략히 설명하였다. 기술별 자세한 적용 사례는 본고의 범위를 넘어감으로 저자의 다른 자료[20]를 참고하길 바란다.

V. 결 론

본고에서는 HTML5 mobile web application platform 구조에 대해서 mobile OS에 포함되어 배포되는 native부분과, HTML5 application framework부분으로 나누어 상세히 살펴보았다. 그리고 이러한 platform에서 동작하는 Web application의 성능 최적화시 활용 가능한 각종 practice 및 open source tool을 소개하였다. 본고는 mobile용 HTML5 application platform 또는 application 개발시 부디칠 각종 기술 이슈에 대해서 폭넓게 이해를 하고 실무 개발에 적용 가능한 reference로 활용되기를 기대한다.

참 고 문 헌

[1] ABI Research "2.1 Billion HTML5 Browsers on Mobile Devices by 2016" (http://www.gartner.com/it/page.jsp?id=1622614) July 22, 2011.

[2] Kevin Benedict, "Interview: Sanjay Poonen on SAP's Mobile Strategies", April 2012 (http://wireless.sys-con.com/node.2261851).

[3] Gartner Press Release (http://www.gartner.com/it/page.jsp?id=1622614).

[4] HTML5, W3C Editor's Draft, June 2012 (http://dev.w3.org/html5/spec/).

[5] Michael Smith, "The Web Platform: Browser technologies" (http://platform.html5.org).

[6] WebKit, open source Browser 엔진 개발 과제 (http://www.webkit.org).

[7] jQuery Mobile (http://www.jquerymobile.com).

[8] Sehcha (http://www.sencha.com).

[9] AppMobi (http://www.appmobi.com).

[10] DOM, Document Object Model (http://www.w3.org/DOM) W3C 표준 specification.

[11] Dominique Hazael, "Standards for Web Applications on Mobile: current state and roadmap" (http://www.w3.org/2012/05/mobile-web-app-state).

[12] jQuery, (http://www.jquery.com).

[13] jQuery Mobile Theme Roller, (http://www.jquerymobile.com/themeroller).

[14] jQuery UI, (http://www.jqueryui.com).

[15] PageSpeed Tools, (http://developers.google.com/speed/pagespeed).

[16] DOM Monster, (http://miraculo.us/dom-monster)

[17] Chrome Developer Tools (http://developers.google.com/chrome-developer-tools/docs/overview) Google.

[18] W3C Navigation Timing specification (http://www.w3.org/TR/navigation-timing) July 16th 2012.

[19] 허준회, "HW Acceleration in WebKit" (http://www.slideshare.net/joone/hardware-acceleration-in-webkit).

[20] 임상석, "Mobile Web Application 개발 방법론: Open Source 기반 Web App 개발 Practice",(http://www.slideshare.net/infect2/web-app-201205).

[21] Speed Tracer (https://developers.google.com/web-toolkit/speedtracer) Google.

[22]Closure Compiler (https://developers.google.com/closure/compiler/) Google.

[23]CSS Sprite Generator (http://csssprites.com) Google.

약 력

임 상 석

1998년 전남대학교 공학사
2000년 한국과학기술원 공학석사
2006년 한국과학기술원 공학박사
2004년~2005년 IBM T.J Watson Research Center, Intern 연구원
2006년~2011년 삼성전자 DMC 연구소, 책임 연구원
2011년~현재 SK 플래닛 플랫폼, 기술원 매니저
관심분야: Web platform, WebKit, JavaScript, open source, mobile browser