

WebComponents 기반의 웹 플랫폼 개발: 텍스트큐브 2

신정규 / 래블업
lablup.com



■ 신정규

- 복잡계 이론 물리학자

- 통계물리학 / 뇌과학 / 뇌 정보 네트워크 전공

-  래블업

- 코딩 교육 시스템 / 연구 지원 시스템 (코드온웹)
 - 오픈소스 기계학습 분산처리 시스템 (소르나 프레임워크)

-  텍스트큐브

- 오픈소스 블로그 소프트웨어
 - TNF/ Needworks

- 

- 자동차 시스템 보안



그럼 갑시다!

인생은 카오스지만
아기들은 귀여워요



■ 문제의식

■ 문서? 런타임 플랫폼?

- Mozilla의 접근
- Google의 접근
- Apple의 접근
- Facebook의 접근

■ 충돌

- WHATWG (web hypertext application technology working group)



■ 충돌: 종교 전쟁의 시작

- W3C
 - Next HTML specification
- WHATWG (web Hypertext application technology working group)
 - Web application 1.0 (2004. 7)
- 2007 ~ 2012
 - HTML5로 리브랜딩 및 규격 확정 작업에 동의
 - HTML5에 대한 양 집단의 “정의 (definition)” 가 다르다



More recently, the goals of the W3C and the WHATWG on the HTML front have diverged a bit as well. **The WHATWG effort** is focused on developing the canonical description of HTML and related technologies, meaning fixing bugs as we find them **adding new features as they become necessary and viable**, and generally tracking implementations.

The W3C effort, meanwhile, is now **focused on creating a snapshot** developed according to the venerable W3C process. This led to the chairs of the W3C HTML working group and myself deciding to **split the work into two, with a different person responsible** for editing the W3C HTML5, canvas, and microdata specifications than is editing the WHATWG specification.

Ian Hickson (WHATWG editor)



■ 불안한 동거 (가내 별거)

- 문서다 (W3C)

- 고정된, 한정된 형식과 포맷
- 영속성에 중점
- 앱은 앱으로 만들지 왜 웹 위에 만드냐?

- 플랫폼이다 (WHATWG)

- 변화하고 발전하는 규약
- 확장성 및 진화에 중점
- 너희는 우리 규격의 스냅샷일 뿐



■ 현재

- 기업이 개발하는 브라우저
 - Chrome / Edge
 - 새 규격을 계속 새 브라우저에 부어 넣음
- 기업 외 브라우저
 - Firefox
 - 규격을 추가하고 있긴 하지만 여전히 의문 제기
 - XUL 기반의 웹 플랫폼을 말아먹은 전력이 있음
- 이상한 놈
 - Safari
 - W3C만 지원할까?+독자 명령셋 (--webkit-) 지원
 - 모바일 왕



■ 런타임 플랫폼적 요소들 (현재편)

- (WHATWG의 HTML5에 추가된) 플랫폼적 요소들
 - HTML Import
 - 외부 HTML을 참조할 수 있음
 - Shadow DOM
 - HTML의 서브셋이 독립된 DOM을 가질 수 있음
 - Custom Elements
 - 브라우저 기본 DOM의 leaf object /class 를 개발자가 임의로 만들 수 있음
 - Templates
 - 템플릿을 만들고 디자인할 수 있음
 - CSS Containment와 잘 어울릴겁니다



■ 런타임 플랫폼적 요소들 (미래편)

- WebUSB / Web Bluetooth / Web MIDI
 - 브라우저가 동작중인 기기에 달린 하드웨어 포트를 제어할 수 있음
- Native Clients
 - 생각하시는 그거 맞습니다. (샌드박스 안에서 돌아감)
- Window-controls
 - 웹앱에서 윈도우 컨트롤 요소 제어
- DDN (Device discovery notification)
 - 주변 기기 탐색 및 알림



- WebGL
 - OpenGL ES 2기반의 웹에서 그래픽 가속 사용을 위한 라이브러리.
- WebAssembly
 - 바이너리 포맷의 모듈을 마치 JavaScript의 네이티브 모듈처럼 사용
- WebPayments
 - 웹에서 결제 과정에 해당되는 부분의 규격
 - 사용자 – ISP – 서비스 간의 통신에 대한 표준 제공
 - 덧) 상용화는 사과사가 시작…



- Streams

- Multiple Fetch -> Process-> Render 루트

- RequestIdleCallback

- 중요하지 않은 일들을 Idle때 처리할 수 있는 callback

- Passive event listener

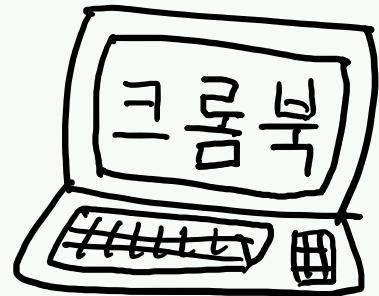
- 이벤트 리스너도 패시브하게 동작할 수 있게 하는 옵션
 - JavaScript의 thread bottleneck 을 해결하기 위해 등장

- ResourceHints

- <link rel="preload / prefetch / preconnect">
 - 특정 링크는 백그라운드에서 미리 프리패칭하도록 알려줌



- **ImageBitmap**
 - Service Worker와 연동해서 이미지를 생성 및 처리할 때 유용
- **MediaRecorder**
 - WebRTC에서 음성 / 영상을 직접 녹화해서 보여줄 수 있음 (서버사이드 없이)
- **CSS Variable**
 - CSS 단에서 변수를 지원
 - setDocumentVariable과 함께 사용



Web Components

HTML5 앱의 기어박스







■ HTML Import

- 임의의 HTML 파일을 CSS 불러오듯 import 할 수 있음
- 반복되는 코드 블럭 조각을 불러오는데도 쓸 수가 있지만, custom element 등을 불러오는 용도가 적당합니다.

```
<link rel="import" href="name-card.html">
```



■ Shadow DOM (V1)

- DOM Tree 하위의 어떤 leaf 엘리먼트 안에 독립된 DOM Tree를 만들 수 있음
- Root DOM (우리가 일반적으로 생각하는 DOM) 에서 보이지 않는다고 해서 Shadow DOM
 - 예) Root DOM와 Shadow DOM 내의 같은 id가 중복되어도 상관이 없음
- Chrome만 지원 중. 메이저 브라우저들은 2016년 내에 움직임이 있을 것으로 예상. (그런데 어쩐지 안 움직인다...)

```
this.createShadowRoot().appendChild(new element);
```



■ Custom Element

- DOM에서 HTML5 규격에 미리 정의된 object type 이외의 타입을 생성하고 사용할 수 있음
 - 예) <div>, <a> 태그처럼 <name-card>라는 custom element를 만들어 사용할 수 있습니다.

```
document.registerElement('name-card', {  
    prototype: prototype  
});  
var ncElement = document.createElement('name-card');
```



■ HTML Template

- HTML에서 디자인 요소를 분리해 낸 것이 CSS
- 구조를 분리해 낸 것이 Template

```
<style>
  .card {
    width : 200px;
    height: 35px;
    border-radius: 3px;
  }
</style>
<template id="namecard-root">
  <div class="card">
    <h2>Name : <span>{{ name }}</span></h2>
  </div>
</template>
```



두근

두근

와작!

와작!

■ Shadow DOM (V1)

- DOM Tree 하
Tree를 만들 수
없는 경우
- Root DOM (원
 않는다고 해서
■ 예) Root DO
상관이 없음
- Chrome만 지
움직임이 있을



this.createShadowRoot();

new Element());

설정된 DOM

)에서 보이지

등 복되어도

6년 내에

Webcomponents.js

기다리다 지친 그대들을 위해

<http://webcomponents.org>



■ Webcomponents.js

- “Polyfill”
 - HTML5 webcomponents 스펙을 완전히 지원하지 않는 브라우저들을 위한 대체 구현
- Webcomponents.js
 - Custom elements / HTML imports / Shadow DOM
 - Mutation observers
 - DOM mutation 을 추적하고 효율적으로 실행함
 - ES6 Weakmap type
 - Key 를 지정하지 않은 k-v Map



■ 브라우저 지원

Polyfill	IE10	IE11+	Chrome*	Firefox*	Safari 7+*	Chrome Android*	Mobile Safari*
Custom Elements	~	✓	✓	✓	✓	✓	✓
HTML Imports	~	✓	✓	✓	✓	✓	✓
Shadow DOM	✓	✓	✓	✓	✓	✓	✓
Templates	✓	✓	✓	✓	✓	✓	✓

- Webcomponents.js : 모든 polyfill 지원
 - 115kbytes
- Webcomponents-lite.js : Shadow DOM 지원 제외
 - 38kbytes



■ 준비

- 코드 한 줄 추가!
- <script src="webcomponents-lite.min.js"></script>
- 이제 본격적으로 떠나볼까요?



사전지식 3:

Polymer

<http://polymer-project.org>

구글의 대답



■ Polymer library

- HTML5 컴포넌트 기반 라이브러리
 - HTML imports / Custom elements / Shadow DOM
- Web components에 기반함
 - “Polyfill” 을 지원 : 브라우저 호환 레이어
- 재사용 가능한 컴포넌트 개발
 - ‘DOMelements’라는 이름 사용
 - 원래 이름으로는 표준화하기 좀 그랬던듯
 - (PolymerElement)



여러분은 이미
폴리머를 본 적이 있습니다. ...진짜임



■ Polymer: 요약

- Polyfill library + MORE
 - 브라우저가 안 되는 부분을 메꿔줍니다.
- Template + style + code로 딱딱 나누어짐
- Two-way binding 지원 (느림..)
- 컴포넌트 상속 지원
 - 정확하는 상속이라기보다는 메소드 바인딩
- 직관적인 구조



■ Polymer : 장점

- 올인원 솔루션
 - 이것저것 다 구현해 놓았음
- Webcomponents.js에 지대한 공헌
 - 사실상 멱살 잡고 끌고 간다고 봐도 과장은 아님
- Component-driven
 - customelements.js 등에서 공유되는 custom elements들이 많음
 - 그냥 가져와서 쓰면 된다
- 크롬 브라우저의 네이티브 지원
 - Polyfill이 필요하지 않다! (크롬에서만)



■ Polymer : 단점

- 엄청난 덩치
- 느림
- DOMElement 의존성 체크 오버헤드
 - Vulcanize : 미리 몽땅 체크해서 중복 import 구조를 없애고 파일 하나로 통합 - 엄청 커집니다
 - Crisper : CORS 이슈 해결을 위해 Javascript를 HTML에서 뜯어냄
- 불안정
 - 디폴트 컴포넌트로 주는 컴포넌트들의 변화가 심함



■ 관련 접근

- Progressive Web App (2016년 4월)
 - 웹을 앱처럼 사용하자!
 - Node.js에서 일반적인 접근인 REST의 개념에 Service Worker로 오프라인 지원을 추가
- PRPL Pattern
 - 구글의 제안
 - 웹페이지의 객체+스트리밍화



■ PRPL 패턴

- 웹페이지를 구성하는 컨셉의 변경
- 처음엔 엄청 중요한 컴포넌트들만 **푸시**하고
- 최대한 빨리 첫 루트를 **그려서** 보여준 다음
- 나머지 루트들은 **미리 캐싱하고**
- 요청에 따라 **필요할 때 로딩**하고 다음 루트를 만들어 보여준다



■ 준비물

- PRPL을 가능하게 만드는 기술들
 - Web components 지원 플랫폼
 - HTTP/2 Server-push 지원 웹서버
 - Service Worker 지원 웹 브라우저





*Image by co.milesplit.com



텍스트큐브 2

폴리머로 갈아 엎기



■ 텍스트큐브

■ 텍스트큐브: Brand Yourself!

- 2006년부터 시작한 온라인 출판 플랫폼 (블로그)
- 국내에서 최초로 자생한 대규모 오픈소스 프로젝트
- 개인 웹로그 부터 블로그 서비스 까지 운영 가능한 도구
- 커뮤니티 개발 오픈소스 소프트웨어

■ 태터툴즈

- 텍스트큐브의 전신
- 개인 개발 프로젝트 (JH님) (2004), 회사 설립 (2005), 오픈소스화 (2006)



- PHP / Apache · Nginx · IIS / MySQL · MariaDB · PostgreSQL · Cubrid · SQLite 지원
- 유연한 스킨 / 플러그인 아키텍처
- RDF / BlogAPI / OpenAPI / Microformats / XFN / FOAF / geolocation (if browser supports) 지원
- 키워드 로그 / 지역 로그 / 댓글 알림 / 모바일 폰 및 이메일 블로깅 / 팟캐스팅 / RSS 리더 / 마이크로 블로깅 기능
- 오프라인 실행 지원 (베타)



■ 파생 서비스

- 티스토리 (tistory.com)
 - TNC+Daum
 - 태터툴즈 멀티유저 서비스 모드에서 브렌칭 (1.0.5 이후)
 - 소스 트리 동기화를 통해 기능 및 UI 공유 (~2009)
- 블로거 닷컴 (blogger.com)
 - TNC+Google
 - 텍스트큐브 1.6에서 브렌칭
 - textcube.com 으로 시작해서 구글의 TNC 인수 이후 블로거닷컴에 통합



■ 텍스트큐브 2에서 해결할 점

- 복잡한 자바스크립트 라이브러리 / 번들
 - jQuery / Dojo / mootools / lodash / EAF
- 통일성이 적은 UI 관련 자바스크립트 코드
- 근본론자의 코드
 - XHTML1은 지켰으나 시대에 늙어버림
 - 'javascript를 꺼도 돌아가는 플랫폼' 이 지금 와서 의미가 있는가?



■ Polymer 도입

■ Polymer를 선택한 계기

- 익숙함: (비교적) 오래 다루어옴
- React에 대한 거부감
 - 아직까지 JSX는 차마 손을 못대겠음 π_π
 - 내용과 코드를 어찌 섞는단 말인가!
- 도전
 - jQuery는 너무 오래 해 왔으니까…



■ 도입예

■ 레이아웃 재작성

- <https://elements.polymer-project.org/browse?package=app-elements>
- App-layout 기반으로 재작성

```
<body id="body-entry">
  <div id="tcDialog" style="display:none;"></div>

  <app-drawer-layout id="app-body" responsive-width="900px" drawer-width="140px">
    <app-drawer swipe-open class="drawer-menu">
      <app-header id="menu-textcube" waterfall fixed>
        <div id="menu-textcube"><a href="/textcube/trunk/?owner=center/dashboard" title="센터로 이동합니다">서브메뉴 : 글</a>
      </app-header>
      <h2>서브메뉴 : 글</h2>

      <div id="sub-menu-box">
        <ul id="sub-menu">
          <li id="sub-menu-entry" class="selected firstChild"><a title="글">글</a>
          <li id="sub-menu-line"><a title="라인 관리" href="/textcube/trunk/?owner=center/line">라인 관리</a>
          <li class="divider"><span class="divider">-</span></li>
          <li><a href="#">다음 메뉴</a>
        </ul>
      </div>
    </app-drawer>
  </app-drawer-layout>
</body>
```

■ 도입예

- <app-drawer-layout>
 - <app-drawer>
 - <app-header>
 - <app-header>
 - <app-header>-layout
 - <app-header>
 - <app-toolbar>

```
<body id="body-entry">
  <div id="tcDialog" style="display:none;"></div>

  <app-drawer-layout id="app-body" responsive-width="900px" drawer-width="140px">
    <app-drawer swipe-open class="drawer-menu">
      <app-header id="menu-textcube" waterfall fixed>
        <div id="menu-textcube"><a href="/textcube/trunk/?owner=center/dashbo</div>
      </app-header>
      <h2>서브메뉴 : 글</h2>

      <div id="sub-menu-box">
        <ul id="sub-menu">
          <li id="sub-menu-entry" class="selected fi</li>
          <li id="sub-menu-line"><a title="라인 관리"></a></li>
          <li class="divider"><span class="divider"></span></li>
          <li id="sub-menu-post"><a title="글 쓰기" href="/textcube/post?owner=center/dashbo</li>
          <li id="sub-menu-page"><a title="페이지 만들기" href="/textcube/page?owner=center/dashbo</li>
          <li id="sub-menu-notice"><a title="공지 쓰기" href="/textcube/notice?owner=center/dashbo</li>
          <li id="sub-menu-keylog"><a title="키워드 만들기" href="/textcube/keylog?owner=center/dashbo</li>
          <li id="sub-menu-template"><a title="서식 만들기" href="/textcube/template?owner=center/dashbo</li>
          <li class="divider"><span class="divider"></span></li>
          <li id="sub-menu-category"><a title="분류 관리" href="/textcube/category?owner=center/dashbo</li>
          <li id="sub-menu-tag"><a title="태그 관리" href="/textcube/tag?owner=center/dashbo</li>
        </ul>
        <div id="custom-sub-menu">
          </div>
        </div>
      </div>
      <div id="main-description-box">
        <ul id="main-description">
          <li id="description-blogger"><span class="text">한국어</span></li>
        </ul>
      </div>
      <div id="main-blog-box">
        <div id="main-blog">
          </div>
      </div>
    </app-drawer>
    <app-header-layout main id="main-panel">
      <app-header id="main-toolbar" condenses reveals
        effects="waterfall parallax-background" style="height: 35px; width: 140px; position: absolute; top: 0; left: 0; z-index: 1000; background-color: #fff; border-bottom: 1px solid #ccc; transition: all 0.3s ease-in-out; transform: translateZ(0);>
        <div id="main-menu-box">
          <ul>
            <li id="menu-center"><a href="#">메뉴</a></li>
            <li id="menu-profile"><a href="#">프로필</a></li>
            <li id="menu-signout"><a href="#">로그아웃</a></li>
          </ul>
        </div>
      </app-header>
    </app-header-layout>
  </app-drawer-layout>
</body>
```

문제점

아마 여기쯤 왔으면 시간이 모자랄거니까…



■ 실무적 문제

- 사파리 / iOS가 Service Worker 를 지원하지 않음
 - 웹킷 implementation list에 아예 고려가 안 되고 있음
 - Pre-cache를 못 쓰므로 PRPL 패턴 적용이 불가능
- 크롬을 제외한 브라우저들이 HTML import를 지원하지 않음
 - ES6과의 기능 충복 문제로 토의중
- WebComponentsReady 이벤트가 Native Web components를 지원하는 크롬에서는 발생하지 않음
 - Polyfill을 해야 하는 브라우저들은 해당 이벤트 이후에만 HTML import / template 등을 쓸 수 있음
 - 따라서 이중 코드를 작성해야 할 일이 생길 수 있음



■ 실무적 문제

- Shadow DOM을 사용할 경우 모바일 플랫폼에서 엄청나게 느림
 - iOS에서 webcomponents.js로 polyfill 한게 안드로이드의 네이티브 shadow DOM 보다 빠름
- Webcomponents.js는 async로 불러올 수가 없음
 - Polyfill에 걸리는 시간이 2년 넘은 모바일 기기들에서는 너무 김
- App-storage의 동작을 예측할 수 없음
 - 크롬에서 '만' 잘 됨
- Django 등의 프레임워크와 함께 사용하려면 고민을 해야 함
 - 템플릿 프리픽스 / 치환자가 겹치는 경우가 있음



■ 해결책

- 사파리 / iOS가 Service Worker 를 지원하지 않음
 - cache.manifest를 이용해서 파일 캐시 사용하는 법이 있음
 - On-demand가 아니지만 없는것보다는 나음
- 크롬을 제외한 브라우저들이 HTML import를 지원하지 않음
 - 현재로서는 polyfill을 이용할 수 밖에 없음



- WebComponentsReady 이벤트가 Native Web components를 지원하는 크롬에서는 발생하지 않음
 - Webcomponents.js 는 Chrome에서는 아예 불러오지 않게 짠 후
 - 크롬에서 동일한 이름의 이벤트에 불지르는 코드를 하나 짜서 넣으면 됨

```
wcReady = document.createEvent("HTMLEvents");
wcReady.initEvent("WebComponentsReady", true, true);
wcReady.eventName = "WebComponentsReady";
window.dispatchEvent(wcReady);
```

- Shadow DOM을 사용할 경우 모바일 플랫폼에서 엄청나게 느림
 - 쓰지 맙시다.
 - 조만간 CSS containment 나오면 디자인쪽은 어떻게 될 겁니다
- Webcomponents.js 는 async로 불러올 수가 없음
 - 이 파일만은 safari 로컬캐시에 넣었다가 빼는 법을 쓸 수가 없음
- App-storage 의 동작을 예측할 수 없음
 - 분기를 타는 식으로 구현
- Django 등의 프레임워크와 함께 사용하려면 고민을 해야 함
 - Django-rest framework을 쓰고, 템플릿은 쓰지 않으면 됩니다



■ 요약

- HTML5 Web Components 개요
- Polymer 소개
- 텍스트큐브 적용 예
- 문제점과 해결책들



감사합니다!



Lablup Inc.

<http://www.lablup.com>

코드온웹: <https://www.codeonweb.com>

