강의 **05**/16:
OSS Web Backends

신정규

2017년 3월 30일

하나씩 하나씩

좀 속도를 늦춰봅시다.

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)

Polymer : 요약

- Polyfill library + MORE
  - 브라우저가 안 되는 부분을 메꿔줍니다.
- Template + style + code 로 딱딱 나누어짐
- Two-way binding 지원 (느림..)
- 컴포넌트 상속 지원
  - 정확히는 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
  - 어느정도 안정화 (1.7) → 곧 2.0이 나옵니다. 망했어요.

X-Tag

모질라 재단도 손에 카드를 들고 있(긴 하)다...

https://x-tag.readme.io

X-Tag

- 모질라 재단의 일부가 참여하여 개발한 web components 라이브러리
  - 모질라가 web components 개념을 좋아했는지는 별개
  - 개발자들이라면 새 제안이 나오면 손 대 보고 싶죠
- Thin wrapper 추구
  - Shadow DOM 및 그에 기반한 template을 사용하지 않음
- Web components를 날것으로 다뤄보고 싶은 경우 괜찮은 선택이 될 수 있음
  - 실제 써먹기는 애매함
  - 자동차를 주문했더니 매뉴얼과 재료가 택배로 온 느낌임
- MS가 서포트 중

React

페북: 우리는 당신들의 주장에 동의하지 않습니다

저흰 심지어 PHP도 자체 컴파일러 만들어서 써요 *HHVM

https://facebook.github.io/react

React : 같은 문제에 대한 완전히 다른 대답

- 뷰에만 특화된 라이브러리
  - Function들은 view에 바인딩되어 있다.
  - MVC의 V.
- 모든게 컴포넌트
  - 컴포넌트를 가볍게 정의하고, 앱은 컴포넌트들의 컴포넌트가 된다.
- Functional style library
  - 룰이나 코딩 컨벤션을 제공하지 않음
  - 맘대로 써라. (페북은 flux를 적용해서 씁니다)

React : 특징

- 템플릿과 컴포넌트가 완전히 믹스됨
  - 근본론자들은 기겁할 코드
  - 그럴 필요가 없지 않나?

React : 특징

- JSX: 자바스크립트와 HTML의 믹스

```
var nameCard = React.createClass({
    …
    render: function() {
      return (
       <div class="card">
         <h2>Name : <span>{this.state.name}</span></h2>
       </div>
      )
    }
  });

  React.render(
    <nameCard name={userName} />,
    document.querySelector('#name-section')
  );
```


React : 특징

- Fake/Virtual DOM
  - 브라우저의 DOM element의 비정상적으로 거대한 속성 상속 레거시에서 벗어남
  - 빠르다!
- Shadow DOM이 아니다.
  - 실제 DOM이 shadow DOM의 루트 역할에 해당됨
  - Virtual dom은 인스턴트하게만 존재

React : 특징

- 뮤테이션 구현을 할 필요가 없음
  - DOM mutation : 동적 페이지 갱신의 필수 요소
  - 속도 문제는 대부분 여기서 발생함
    - 대부분의 라이브러리들이 권장하지 않…
    - 지만 Polymer만 하더라도 dom-if 나 two-way data binding, updateStyles() API를 지원함 (느림)
  - Virtual DOM에 diff 적용이 됨
    - 바로 변경 사항 적용이 됨
  - 추가 장점: 밖에서 보면 immutable element처럼 보인다

React : 단점

- 스타일링 구현에 대한 불만
  - 스타일이 컴포넌트 밖에 있어야 함
  - 장점으로 보는 입장도 있지만, 재사용 가능한 엘리먼트라는 측면에서는 단점이라고 보는 것이 타당함
- React 의존성
  - React component는 react 안에서만 사용 가능함
- 모바일 환경에서는 일반 javascript 구현보다 느림

Dive into OSS backends

사실상 현재의 모든 벡엔드들은 오픈소스입니다.

백엔드

- 실제 로직이 돌아가는 부분
- Modern (Javascript-based) Platform:
  - 프론트엔드와의 경계가 점차 불명확해짐
- 고전적 웹 백엔드
  - ‘Rendering` - HTML 페이지를 생성해서 브라우저에 전달
  - (Generated) HTML + (Static) CSS

OSS Web backends / frameworks

- Python
  - Django
  - Flask
- PHP
  - Laravel
  - CakePHP
  - Symfony
  - Codeigniter
- Node
  - Express
  - Meteor.js
- Ruby
  - Ruby on Rails
- Java
  - Spark
  - Spring
  - Struts

현대적 웹 백엔드

- API-based 프론트엔드의 태동
- 모바일 앱의 등장

REST Framework (1)

- API
  - Application Programming Interface
  - 운영 체제나 프로그래밍 언어를 위한 인터페이스
- 웹 API
  - 왜? – 클라이언트의 역할이 점차 비대해짐
  - Modern Javascript: 왜 항상 서버를 거쳐 렌더링을 해야 하는가?
- XML to JSON
  - 세상의 변화
  - GET / PUT / POST / DELETE / PATCH

REST Framework (2)

- REST API
  - REST: Representational State Transfer
- 원칙
  - 균일한 인터페이스
  - 상태 없음
  - 캐시
  - 클라이언트 / 서버 구조
  - 계층 시스템
  - 요청에 따른 코드 생성

```
POST
http://example.com/api/add_user/
{
    “username”: {
        “name”: “jkshin”
    }
}
```


REST Frameworks

- Django REST Frameworks
  - REST에 특화된 Django
  - 기존 Django와의 차잇점: View가 API + Serializer로 되어 있음
  - 이후 레거시 Django 와 통합 예정

- Express
  - Node.js 기반의 REST 구현시 많이 사용

Assignment #3

- 노트앱 UI 만들기 (2)
  - 프론트엔드 라이브러리를 하나 골라 UI를 만들어봅시다.
  - “Boilerplate” 로부터 하나씩 해 봅시다.
    - “Boilerplate”: 바로 사용해 볼 수 있을 정도로 준비된 템플릿 예제
  - 스토리보드를 어떻게 적용할 지 고민해 봅시다.
    - Router를 사용해서 SPA로 만들지, 페이지 이동을 구현할지 정합시다
    - 내 스토리보드를 구현하기 위해 필요한 라이브러리들을 조합해봅시다
  - 제출할 것: 만든 UI 가 있는 github 주소 (이번주 일요일 24시까지)

Next is...

6/16: OSS Database management system

@inureyes
Questions? inureyes@gmail.com