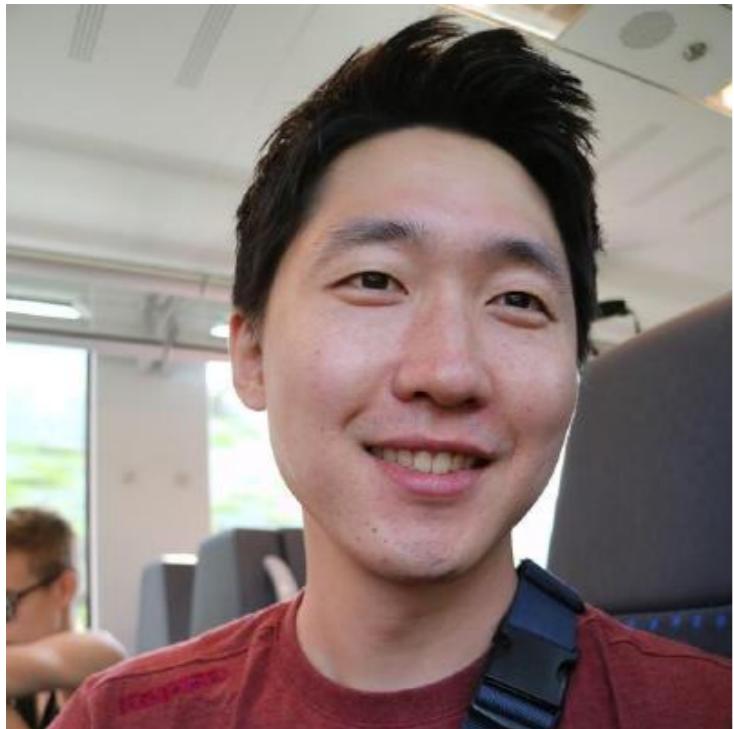


REACT SEOUL

리액트로 다른 페이지에
임베드되는 웹앱 개발기

안재하



안재하

카카오 포털개발팀

github.com/eu81273

programmingsummaries.tistory.com

**리액트로 앱을 만들다보면,
“이 방법이 맞는걸까? 더 좋은
방법은 없는걸까?” 를 고민하
게 됩니다.**

(하지만 대부분은 잘 모르니까 헤맸습니다..)



**앞으로 이어질 내용은
삼질하며 고민한 것들이고
정답이 아닐 수도 있습니다.**

(뜬금없는 도입부..)



카카오에서도 리액트를 많이 사용하고 있습니다.

(사실 제 주변만 좀 찾아봤습니다..)



계정관리

블로그관리 ▾

▷) 10월 초대장이 배포되었습니다.



감성 프로그래밍

programmingsummaries.ti... ▾

쓰기



🏠 블로그관리 홈

☰ 콘텐츠

글 관리

페이지 관리

카테고리 관리

공지 관리

서식 관리

설정

💬 댓글·방명록

댓글 관리

방명록 관리

설정

2017년 11월 2일 통계

오늘 방문자 수

1,635

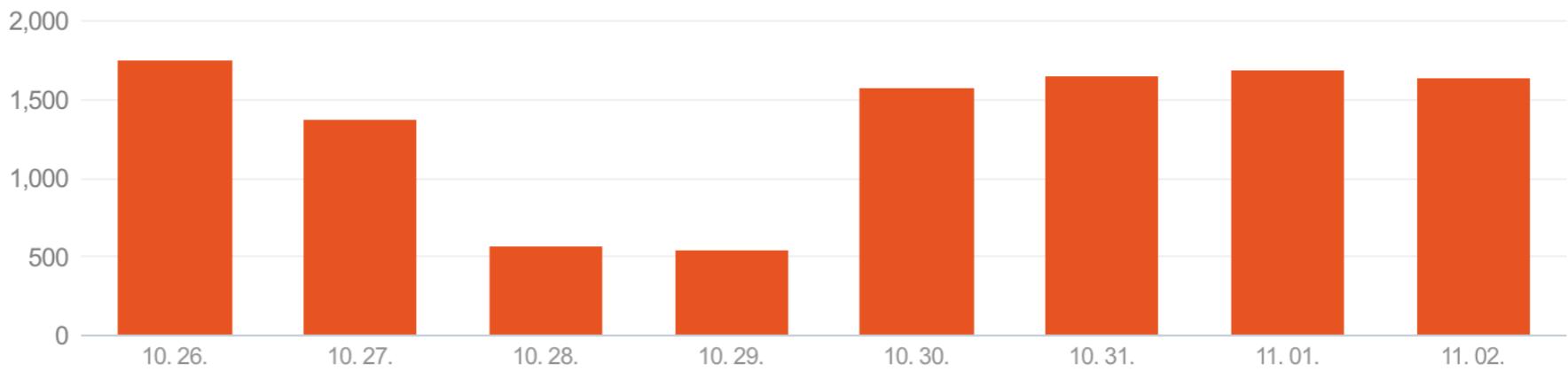
어제 방문자 수

1,680

주간 방문자 수

10,769

전체 방문자 수

1,342,385

유입 키워드 순위

- 1 cmder
- 2 javascript eval
- 3 browserify
- 4 http://programmings...
- 5 javascript useragent

최근 유입 경로

- | | | |
|--|-------------------|---------------------|
| | www.google.co.kr/ | 2017-11-02 22:15:59 |
| | www.google.co.kr/ | 2017-11-02 22:13:17 |
| | www.google.co.kr/ | 2017-11-02 22:13:16 |
| | www.google.co.kr/ | 2017-11-02 22:12:43 |
| | www.google.co.kr/ | 2017-11-02 22:12:27 |

댓글 15

내 댓글

로그인 해주세요

이모티콘



추천순 최신순 과거순



인력장사 그만 해라. 노예 취급 좀 그만해

답글쓰기

47 1



이제라도 돈에 환장한 기업처럼 쓰레기짓 그만하고
국민들에게 사랑 받고 존경 받을수 있는 기업이 되도록 노력들 좀 해라

답글 1

45 0



제빵사 분들이 일자리 줄어들까봐 심히 걱정이 되신답니다.
점주님들은 머하세요? 본사에 압박하셔야죠?
당신들이 직접고용으로 제빵사 인력 줄여서 장사안되면 전액+피해보상 소송하겠다고
빵을 팔아야 되는 빵집이 제빵사가 없어서, 모질라서, 일손이 없어서 빵을 못만드는 사태를
본사가 저질렀으면 본사랑 싸우셔야지 왜 제빵사랑 싸웁니까?

답글쓰기

43 3

더보기 ▼

실시간 채팅

채팅방에는 최대 인원 제한이 있습니다.
입장 가능한 방을 선택해주세요.



채팅방 1 0명

참여하기

여론조사 공표금지기간 이전의 조사결과입니다.

지지율조사

최신조사

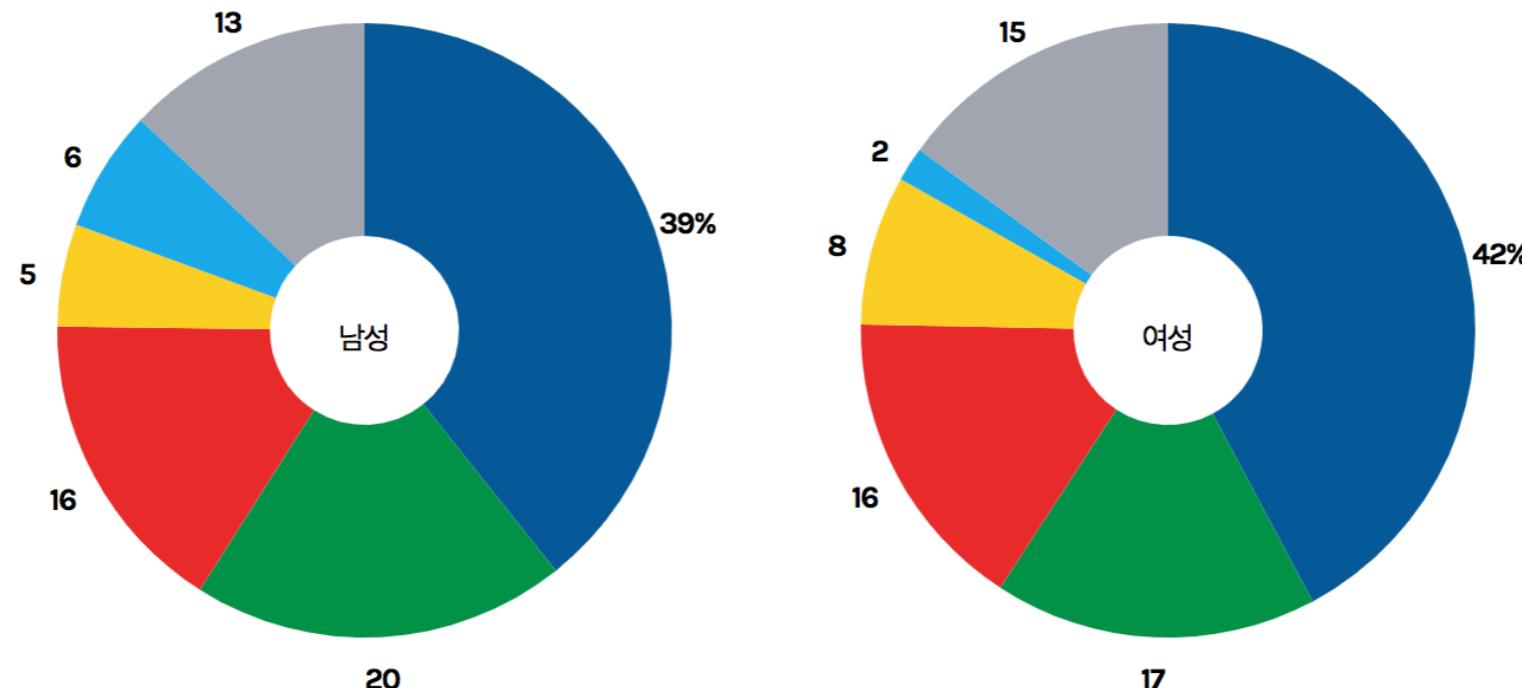
지역별

연령별

성별

미디어별

선거여론 조사결과



- 문재인
- 홍준표
- 안철수
- 유승민
- 심상정
- 기타

SBS · 칸타코리아 여론조사 / 5월1~2일
전국/만19세 이상 남녀 1,023명/유무선 전화면접 RDD/응답률 17.8%/신뢰도 95%±3.0%p

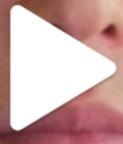
[관련기사 >](#)

'꿀잼' 월화드라마♡ 요즘 가장 빠져있는 작품은?

⌚ 2017.10.24 ~ 2017.10.31

🗳 5,146명 참여중

김재욱·서현진·양세종, 숨막히는 '운명의 삼자대면'



kakaoTV



SBS '사랑의 온도'

02:43

2102명 41%



KBS '마녀의 법정'

03:53

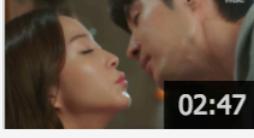
1397명 27%



tvN '이번 생은 처음이라'

03:16

1122명 22%



MBC '20세기 소년소녀'

02:47

525명 10%



투표가 종료되었습니다

**대부분 리액트하면 싱글 페이지
웹앱만 언급되고, 임베드 되는
웹앱은 잘 얘기가 없는 것 같습
니다.**

(사실 리액트를 떠나서 임베드되는 웹앱 자체에 관심이 없..)



사실... 별 차이가 없습니다..

(대신 몇 가지 주의할 점이 있습니다 -.-)



요구사항



**기획 단계 요구사항은,
다양한 종류의 투표 뷰를 가진
임베드되는 투표 웹앱!**

(바 타입/이미지 타입/동영상 타입 등)

**그리고,
한 페이지에서 여러 개가
임베드되어 돌아가면 좋겠다.**

(그리고 결과 뷰를 활용해서 차트도 좀 뿌리면 좋겠다.)



**그리고 당연히(?)
IE8에서 동작해야 한다.**

(인공지능이 바둑두는 시대에 IE8 이라니..)



개발자의 소박한 바램..
리액트 + 리덕스로 만들고 싶다.
차트는 D3 쓰고 싶다.

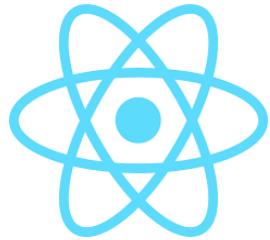
(그리고 이 소박한 바램은 받아들여졌습니다..ㅠㅠ)

IE8에서 돌아가는 리액트

(피할 수 있으면 피하고 싶다..)

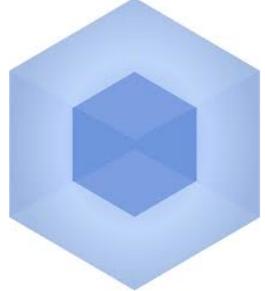
**인터넷 검색을 해보면 IE8에서
동작하는 리액트 앱을 만들기
위한 좋은 글들을 많이 찾을 수
있습니다.**

(그 분들도 이 좁고 험한 길을 걷고 계신거죠..)



React v0.14.9 IE8 을 지원하는 마지막 버전

(하지만 매우 빠르게 mount/unmount를 반복하면 IE의 경우 문제가 생김.. 안습..)



Webpack v1.15.0 IE8 을 지원하는 마지막 버전

(물론 2.0 이상의 웹팩도 번들링에 큰 문제는 없지만..)

BABEL

Babel v6.x.x

**Babel v7 부터는 웹팩 v2 이
상에서 사용 가능**

(이제는 바벨 없이 개발하는건 상상하기 어렵죠..)

```
presets: [
  ['env', {
    targets: {
      browsers: ['ie >= 8']
    },
    loose: true,
  }],
  ['react'],
],
```

IE8 지원을 위해서 webpack 의 babel 설정에서 반드시
loose 모드를 true 로 해야합니다.

```
postLoaders: [  
  {  
    test: /\.js$/,
    loader: 'es3ify-loader'  
  }  
]
```

es3ify-loader 를 사용해서 결과물에 대해 한번 더 후처리
를 해야합니다.

```
new webpack.optimize.UglifyJsPlugin({
  compressor: {
    screw_ie8: false,
    warnings: false
  },
  mangle: {
    screw_ie8: false
  },
  output: {
    comments: false,
    screw_ie8: false
  }
}),
```

UglifyJSPlugin 에서는 반드시 **screw_ie8** 옵션을 **false**로 설정해야 합니다.



**그런데,
뭔가 중요한게 빠진것 같죠?**

폴리필(polyfill)

(사실 이게 중요..)

**ES2015에 추가된 내장 객체와
프로토타입 메서드들은 트랜스파
일링만으로 해결되지 않기 때문에
풀리필이 꼭 필요합니다.**

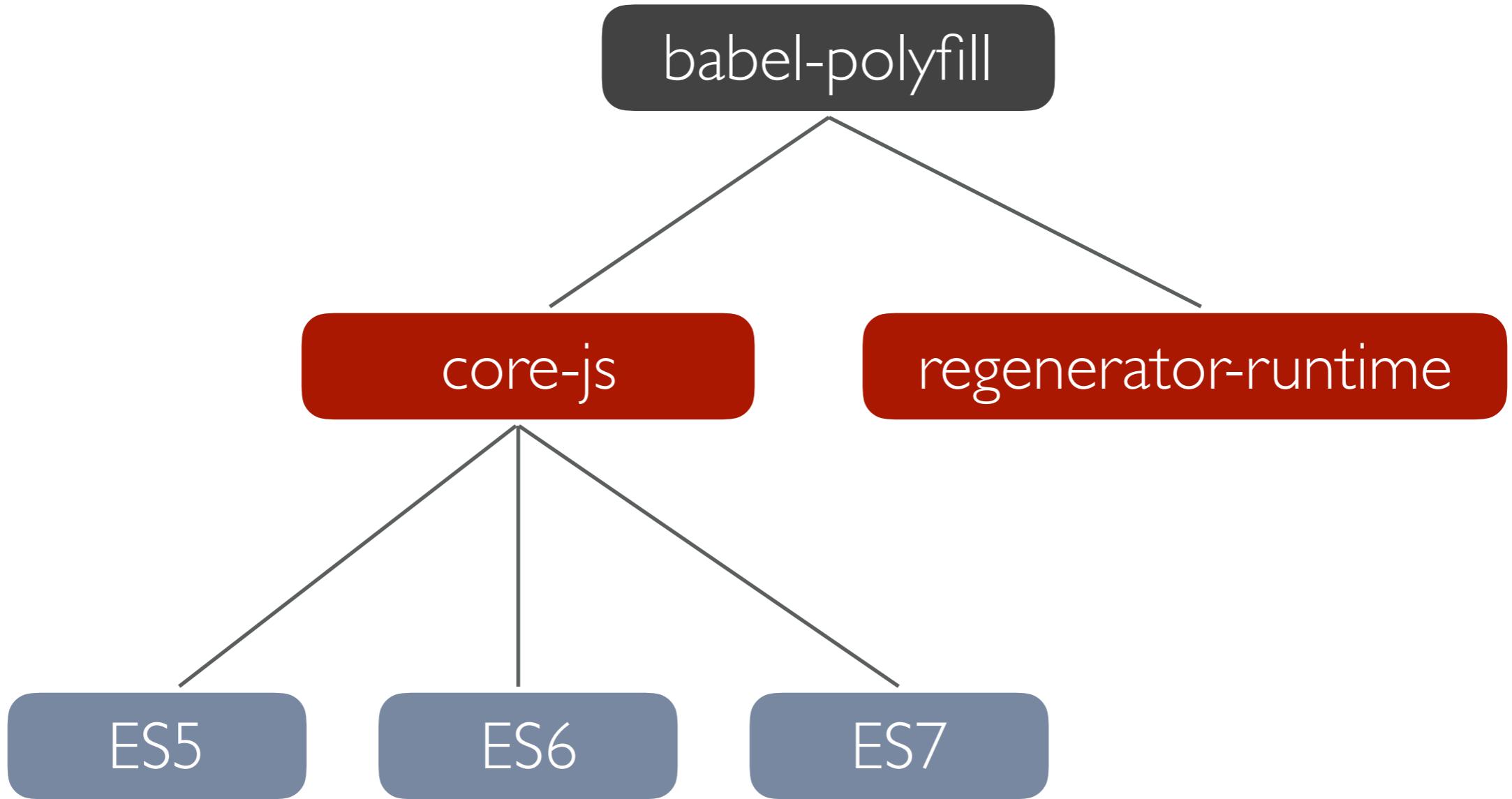
(바벨이 Promise, Map/Set 를 만들어주진 못하기에..)

**그래서 리액트 기반으로 앱을
만들 때 보통은 크게 고민하지
않고 엔트리 포인트의 시작을
이렇게 장식합니다.**

(적어도 저는 그렇습니다..)

```
import 'babel-polyfill';
```

```
(혹은 require('babel-polyfill'); )
```



(주요 폴리필들을 사용하기 편리하게 랩핑해 놓은 모듈..)

babel-polyfill

장점

브라우저에 있는 스펙은 네이티브를 사용해서 성능을 높일 수 있다.

새로 추가된 프로토타입 메서드도 사용할 수 있다.

디펜던시 모듈들이 ES2015+ 스펙을 사용하는지 확인할 필요가 없다.

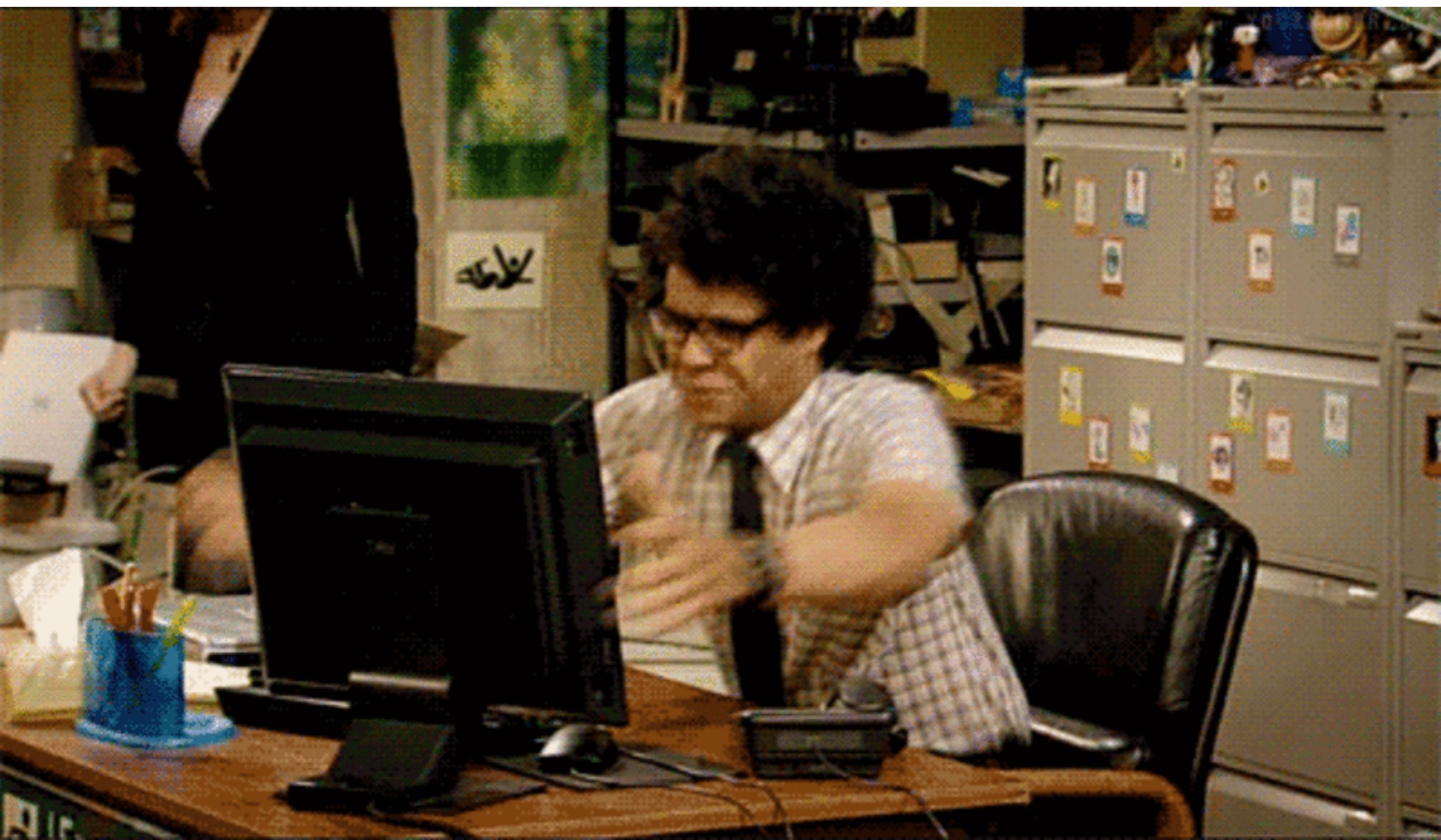
단점

사용하지 않는 폴리필도 모두 추가되므로 번들링되는 파일의 크기 커진다.

전역이 오염된다.

**그런데 임베드되는 웹앱에
babel-polyfill 을 쓰면 어떻게
될까요?**

(답을 이미 아는 분들도 계시겠지만..)



babel-polyfill 은 단 하나의 인스턴스만 허용됩니다.

(쉽게 말하면 두 번 실행하면 안된다는 얘기입니다..)

```
if (global._babelPolyfill) {  
  throw new Error("only one instance of babel-polyfill is allowed");  
}  
global._babelPolyfill = true;  
  
import "core-js/shim";  
import "regenerator-runtime/runtime";
```

(묻지도 따지지도 않고 두 번째 실행하면 무조건 throw..)



**따라서 당연히
나중에 임베드 되는 앱은
실행이 안됩니다.**

(그렇다고 ES2015 를 사용 안 할 수도 없고..)

babel-plugin-transform-runtime

(폴리필 적용을 위한 또다른 옵션..)

```
# babel-plugin-transform-runtime 디펜던시 추가  
$ npm install --save-dev babel-plugin-  
transform-runtime
```

```
# babel-runtime 디펜던시 추가  
$ npm install --save babel-runtime
```

(이렇게 설치 후에..)

```
plugins: [  
  |   | 'transform-runtime'  
  |   | ]
```

(babel-polyfill 과 달리 그냥 이렇게 웹팩 플러그인 설정에
추가해주면 됩니다..)

**폴리필 내장
헬퍼 함수를
참조하도록 변환**

babel-plugin-transform-runtime

**폴리필 내장
헬퍼 함수**

babel-runtime

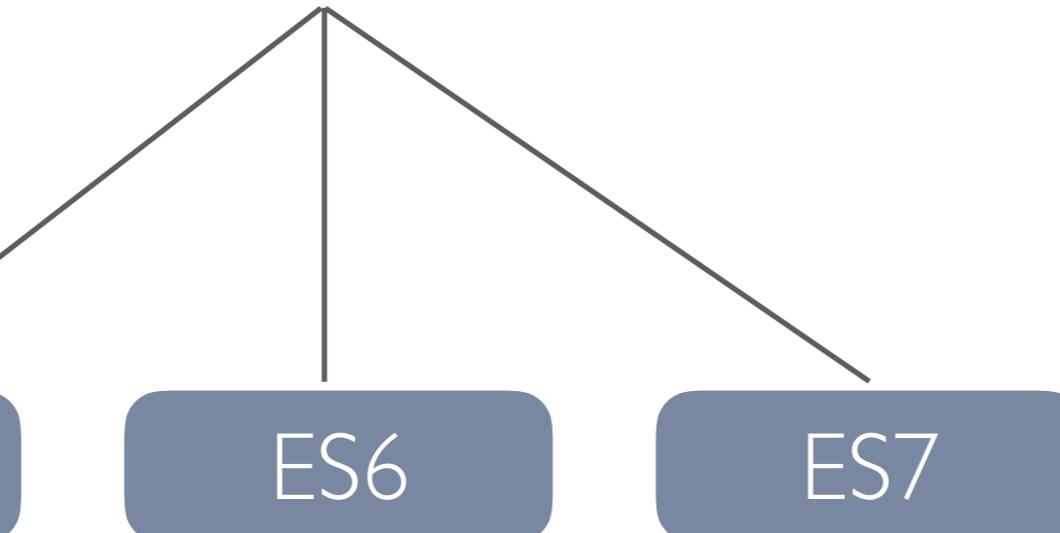
core-js

regenerator-runtime

ES5

ES6

ES7



```
class Person {  
}
```



폴리필이 필요한 부분들을 내장된
폴리필의 alias 를 참조하도록 변경

```
var _classCallCheck2 = require('babel-runtime/helpers/classCallCheck');  
var _classCallCheck3 = _interopRequireDefault(_classCallCheck2);  
function _interopRequireDefault(obj) {  
    return obj && obj.__esModule ? obj : { default: obj };  
  
}  
  
var Person = function Person() {  
    (0, _classCallCheck3.default)(this, Person);  
};
```

babel-plugin-transform-runtime

장점

babel-polyfill 을 사용할 때 보다는
용량이 줄어든다.

전역 오염이 되지 않는다.

단점

새로 추가된 프로토타입 메서드를 사
용할 수 없다.

폴리필 코드가 무조건 물려서 동작하기
때문에 더 느려질 수 있다.

**디펜던시 모듈들이 ES2015+ 스펙을
사용하는지 확인해야 한다.**

```
loaders: [
  {
    test: /\.js|jsx$/,
    exclude: /node_modules/,
    loader: 'babel',
    query: {
```

일반적으로 관례처럼 웹팩 바벨 설정에서 **node_modules** 폴더는 제외하지만,

```
include: [
  /node_modules\axios/
],
```

예를들어 **axios** 처럼 **Promise** 를 사용하는 디펜던시가 있다면,
웹팩 바벨 설정에 함께 트랜스파일링 되도록 추가해야 합니다.

이러면 끝일까?

(끝이면 이렇게 말하지 않았...)

**Babel은 ES2015+ 문법을
ES5 으로까지만 트랜스파일링
해줍니다.**

(그래서 트랜스파일링 해도 IE8에서는 돌릴 수가 없..)

**정말 다행히도 core-js/es5
폴리필은 여러번 중복해서
호출되어도 오류가 발생하지
않습니다.**

(아직 신은 IE8을 버리지 않으셨...)



```
import 'babel-polyfill';  
import 'core-js/es5';
```

(core-js/es5 폴리필은 babel-polyfill 과도 충돌하지 않습니다.)



이제 드디어 임베드되는 웹앱
에 **Hello World** 를 띄울 수 있
게 됐습니다.

CSS

**임베드되는 웹앱에도 CSS 가
당연히 필요합니다.**

(물론 화면이 없으면 필요가 없습니다 -.-)

**처음에는 하나의 CSS 파일로
구성해서 <link> 태그로 삽입해
주는 방식을 사용했습니다.**

(퍼블리셔와 협업에서 유리한 점이 많았지만..)

**늘 그렇듯 시간이 지나면서
CSS 파일을 유지보수하기 어려
웠습니다.**

(나중 되니 어느 부분이 어디에 연관 되는지, 지금도 필요한
건지 파악이..)

그래서 CSS 파일을 모듈화하고 구조적으로 관리하기 위해 SCSS 를 사용했습니다.

(완전히 해소되는건 아니지만 그래도 불필요한 영역을 파악하는데는 좀더 도움이 됐습니다.)

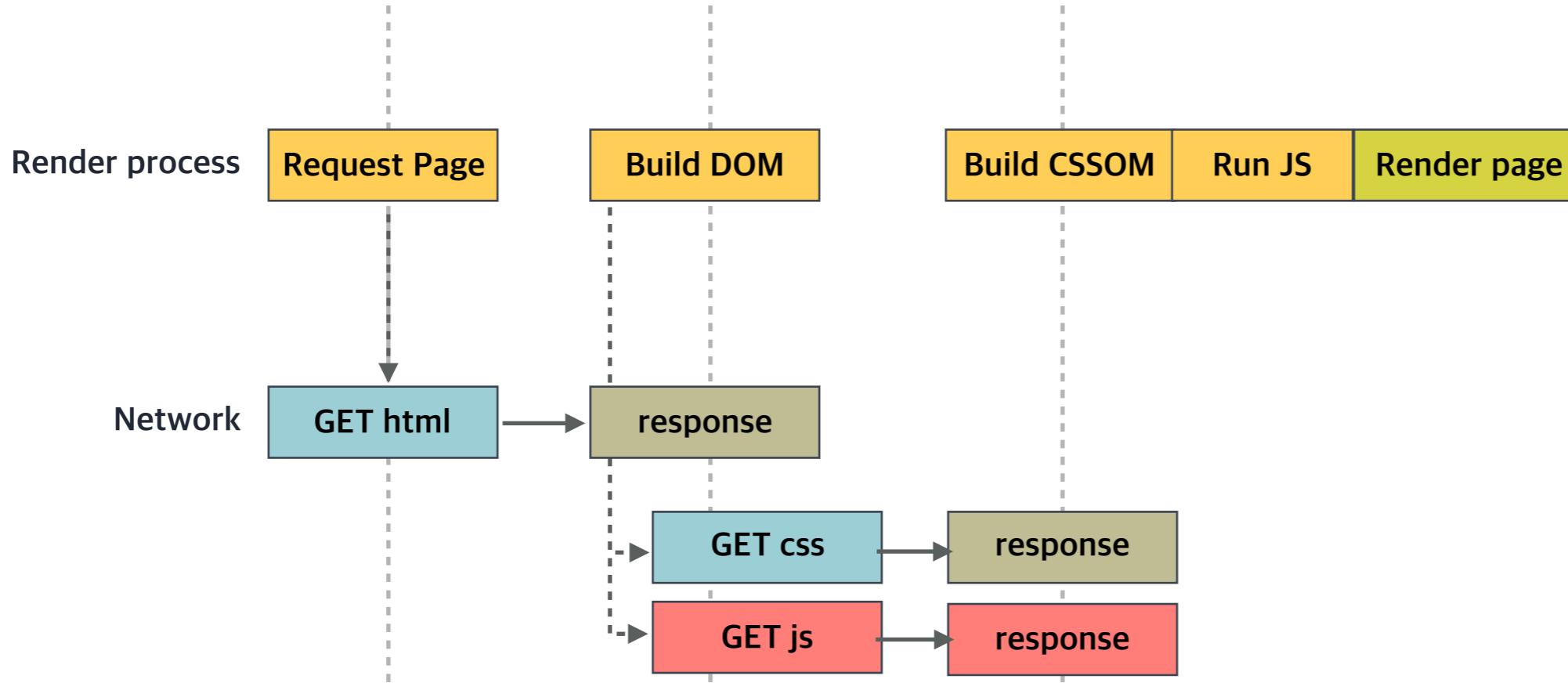
```
@import 'common';
@import 'loading';
@import 'error';

@import 'layers';
@import 'icons';
@import 'poll-header';
@import 'poll-contents';
@import 'poll-buttons';
```

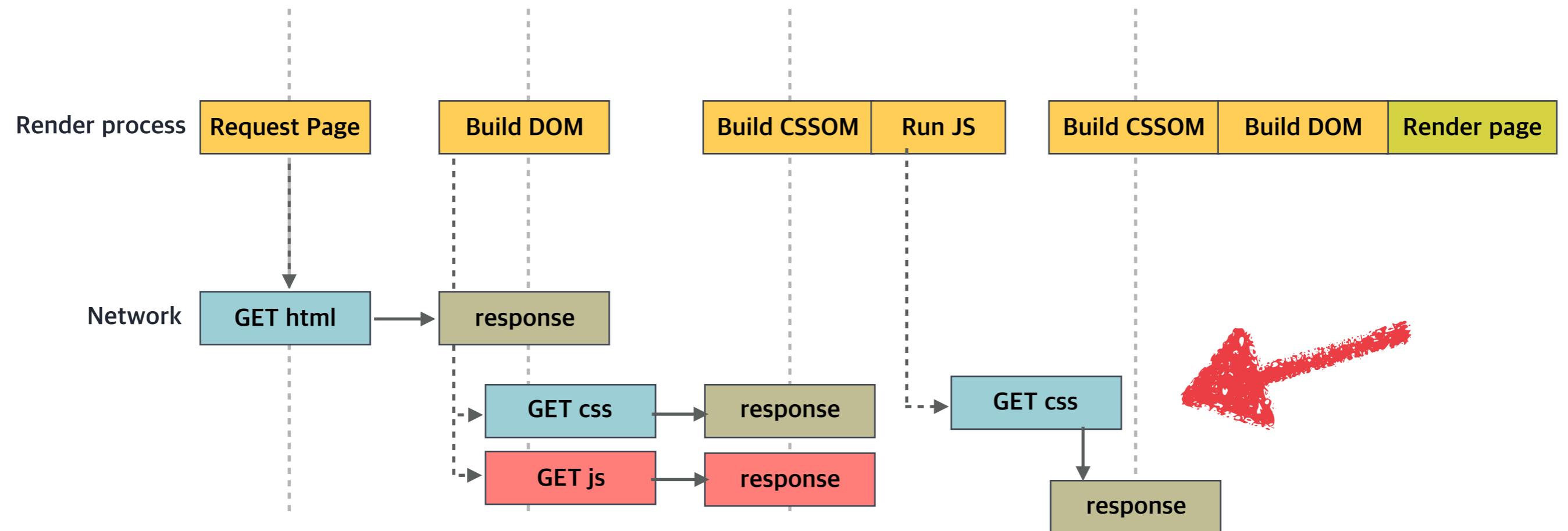
이런식으로 실제 컴포넌트와 어느정도 매칭되도록 모듈화를 했습니다.

**다른 문제는, 로딩 순서로 인해
웹페이지 렌더링에 병목현상을
가져올 때가 많았습니다.**

(크리티컬 렌더링 패스라고 하는 그거 맞습니다..)



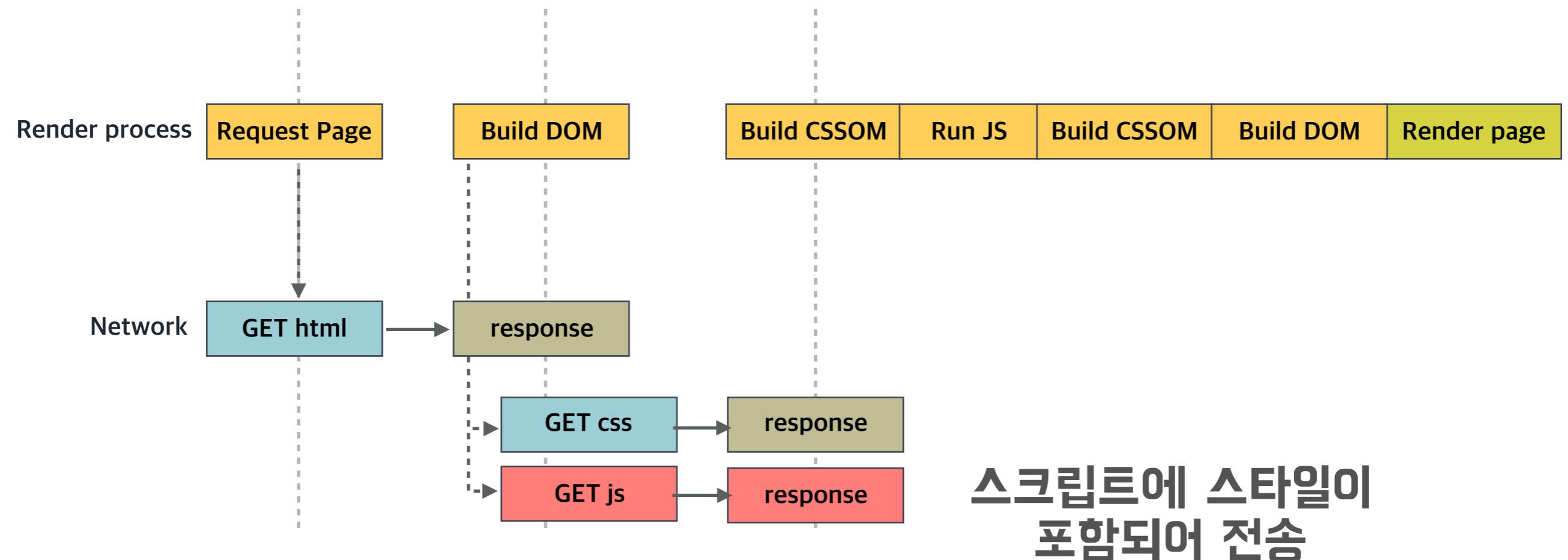
일반적인 웹페이지나, 싱글 페이지 웹앱은 대략 이런 렌더링 경로를 보이게 됩니다.



하지만 임베드 스크립트 로딩 뒤 CSS 파일 주소를 `<link>` 태그로 추가하면 최소 한번의 왕복이 추가됩니다.

**그래서 `css-in-js` 를 도입해서
이 부분을 어느정도 `극복`할 수
있었습니다.**

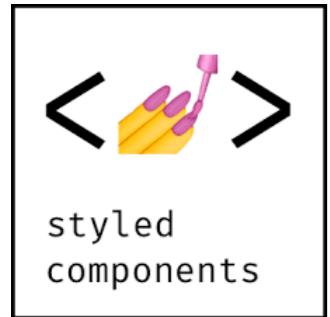
(또다시 익숙한 `css2js` 를 사용해서...)



CSS2JS 를 사용하면 JS와 함께 CSS도 전송되기 때문에
임베드 앱 로딩에 좀더 도움을 줄 수 있습니다.

css2js 를 사용해서 **CSS**를
주입하게 되면 브라우저 캐시로
인해 변경된 스타일 반영에
문제가 생깁니다.

(큰 문제는 아니지만 급하게 변경이 필요할 때 어려움이..)



**사실 *styled-components* 를
사용하면 쉽게 해결됩니다.**

(스타일의 모듈화, 구조화, 스타일 파일 로딩 문제, 브라우저 캐시 문제까지..)

```
import React, { Component } from 'react';
import styled from 'styled-components';

const Title = styled.h1`  
  font-size: 20px;  
  color: #336699;  
`;

class App extends Component {
  render() {
    return (
      <Title>Mystagram</Title>
    );
  }
}
```

styled-components 를 사용하면 CSS를 컴포넌트 레벨
로 격리하고 컴포넌트와 함께 관리할 수 있습니다.

```
.sc-bdVaJa {} .e0FVhe {color: palevioletred; padding: 0.5em 0;}
```

```
.sc-bdVaJa {} .bfMxNZ {color: palevioletred; padding: 1em 0;}
```

```
.sc-bdVaJa {} .e0FVhe {color: palevioletred; padding: 0.5em 0;}
```

클래스명이 무작위로 생성되는 것 같지만,
같은 CSS 속성인 경우 동일한 클래스명이 생성됩니다.

**이제 임베드되는 웹앱에 CSS
까지 적용하게 됐습니다.**

멀티 인스턴스



**임베드 웹앱은 한 화면에 여러 개
가 동시에 구동될 수 있어야 합니다.
따라서 각각 독립된 스크립트에서 동
작해야 합니다.**

(다 그런건 아니고 우리 요구사항이 그랬습니다;;)



이전 다행히 리덕스를 통해 매우 쉽게 해결됩니다.

(그냥 여러개 render 하면 되는거 아닌가? 맞습니다)

```
const containers = Array.from(document.querySelectorAll('[data-app]'));

containers.forEach(container => {
  const dataset = elementDataset(container, 'data-');

  render(
    <Provider store={configureStore({ ...dataset })}>
      <App/>
    </Provider>,
    container
  );
});
```

임베드 앱이 실행되는 엘리먼트마다 단순히 스토어를 새로 생성해서 렌더링해주면 됩니다.

**그런데 앱 인스턴스들이
Serializable 하지 않은 값을
소유해야 할 필요가 있다면 어떻게
해야 할까요?**

(굳이 예를 들자면 내부용 Http Client 클라이언트라던지..)



**리덕스는 Serializable 하지 않
은 값을 리덕스 스토어에 넣는
것을 권장하지 않습니다.**

(물론 자신의 앱이니까 어떻게 쓰든 개발자 맘이라고..)

```
function createThunkMiddleware(extraArgument) {
  return ({ dispatch, getState }) => next => action => {
    if (typeof action === 'function') {
      return action(dispatch, getState, extraArgument);
    }
    return next(action);
  };
}
```

createStore로 스토어를 생성할 때 미들웨어 생성 함수의
파라미터로 extraArgument 주입하는 방식으로 해결합니다.

**이제 임베드되는 웹앱을 여러
개 띄울 수 있게 됐습니다!**

비동기와 미들웨어



처음에는 redux-thunk 로 시작했고, 필요하면 async 액션 생성자 함수를 만들었습니다.

```
export function getAuthAsync(bar) {
  return async function(dispatch) {
    try {
      const foo = await service.auth(bar);
      dispatch(setAuth(foo));
    } catch (e) {
      dispatch(showError(e));
    }
  }
}
```

async function 도 function 이기 때문에 이 정도 수준은 thunk 미들웨어로 감당이 됐습니다.

**async 액션 생성자 안에서
또 다른 async 액션 생성자들
을 불러야 할 필요도 생겼습니다.**

(쪼금 시나리오가 복잡해지기 시작한거죠..)

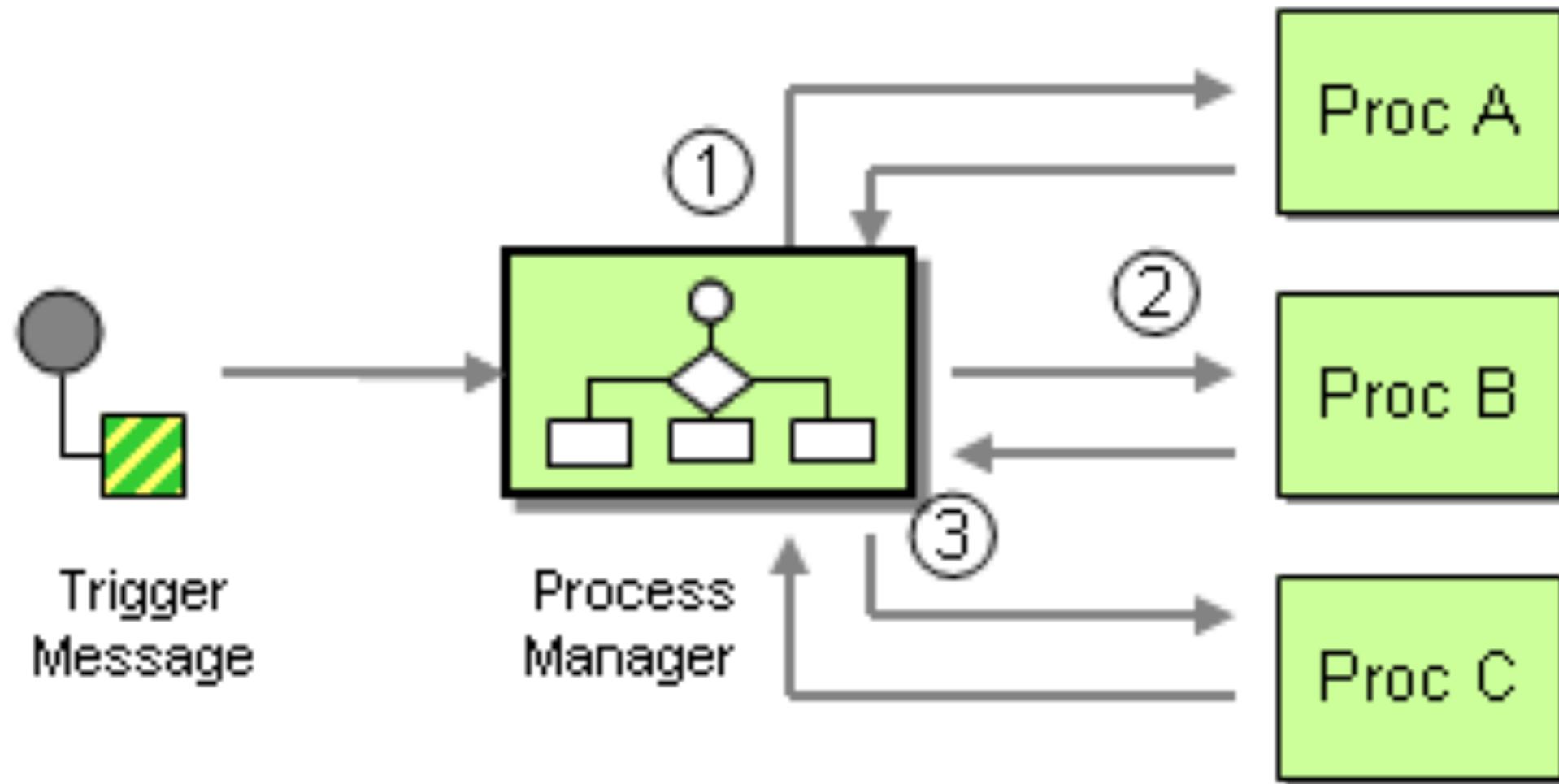
```
function createMiddleware(httpClient, dataset) {  
  return ({ dispatch, getState }) => next => async action =>  
    (typeof action === 'function') ? await action(dispatch,  
      getState, httpClient) : next({...action, dataset});  
}
```

그래서 간단하게 미들웨어를 만들어서 사용했습니다.

```
export function userLogin() { ← 트리거
  return async function(dispatch) {
    try {
      await dispatch(getCredential());
      await dispatch(getUser());
    } catch (e) {
      await dispatch(resetUser());
    }
  };
}
```

서브 트랜잭션
집합

이 미들웨어로 개발을 하다보니 일부 비동기 액션들이 이런 형태를 띄게 되었습니다.



redux-saga 나 **redux-observable** 은 이런 **process manager pattern** 으로 비동기 문제를 접근한다.

라우터



임베드 웹앱은 Browser
History / Location 기반 라
우터는 사용할 수 없습니다.

(너무 당연한 얘기죠..)

이번 앱은 그다지 복잡하진 않아서 리덕스로 scene 이런 상태값을 두어 처리했습니다.

(좀더 규모가 커질수록 내부 라우터 설계가 중요..)

웹앱 인스턴스들 간 상태 공유

(예를 들어 로그인 여부..)

간단한 **pubsub** 으로 처리했는
데 더 좋은 방법이 있을 수 있
습니다.

(일단 저희가 사용하는 범위에서는 괜찮았습니다..)

```
Pubsub.subscribe(USER_LOGIN, async () => {
    await getStorageBearer();
    await getUser();
});

Pubsub.subscribe(USER_LOGOUT, async () => {
    await getCredential();
    await resetUser();
});
```

최상위 컨테이너에서 subscribe 하고 있다가 특정 시그널이 발생하면 액션 생성자를 호출하는 형태입니다.

코딩 스타일

(마지막으로 이건 임베드 웹앱에만 해당되는 건 아니지만..)



**husky + lint-staged +
prettier 적용해두면
커밋할 때마다 자동으로 통일시
켜 줍니다.**

(자연스럽게 코딩 스타일이 통일되는 효과가 있습니다)

```
"scripts": {  
  "start": "./node_modules/.bin/gulp",  
  "test": "./node_modules/.bin/gulp test",  
  "precommit": "lint-staged"  
},  
"lint-staged": {  
  "js/**/*.{js,jsx)": [  
    "prettier --tab-width 4 --trailing-comma all --print-width 100  
    --single-quote --jsx-bracket-same-line --write",  
    "git add"  
  ]  
},
```

이런 느낌으로 precommit 단계에서 prettier 가 실행되도록 합니다.

**제가 준비한 발표는
여기까지입니다!**

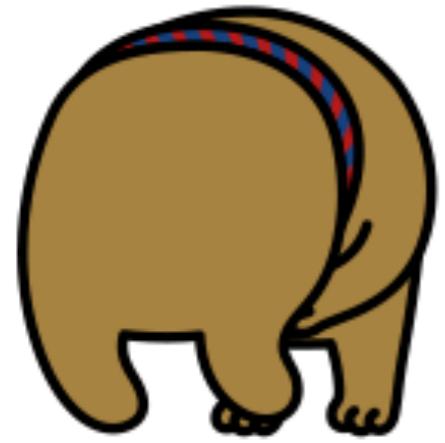
(두서없는 발표 들으시느라 고생 많으셨습니다ㅠㅠ)

결론!
임베드 웹앱, 별 차이 없습니다.
(몇 가지만 주의하면..)



**카카오 포털 개발팀에서
함께 배워가고 함께 성장해 갈
개발자를 찾고 있습니다.**

(많이 지원해주세요!!)



감사합니다.