Ask Company

리액트와 함께 장고 시작하기 / 웹 프론트엔드 기초

JavaScript와 jQuery

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

JavaScript

웹브라우저 내에서 주로 구동되던 프로그래밍 언어

구글이 크롬 브라우저에서 사용되는 V8엔진을 오픈소스로 공개

- 자바스크립트를 실행하기 전에 컴파일한 후에 실행
- 이 V8엔진을 통해 nodejs 플랫폼이 개발됨.

nodejs

범용 JavaScript 소프트웨어 플랫폼

주로 웹서비스 개발에 많이 사용

Electron을 통해 Desktop 애플리케이션 개발도 지원

- https://electronjs.org

jQuery

웹프론트엔드 자바스크립트 계의 혁명이었던 ...

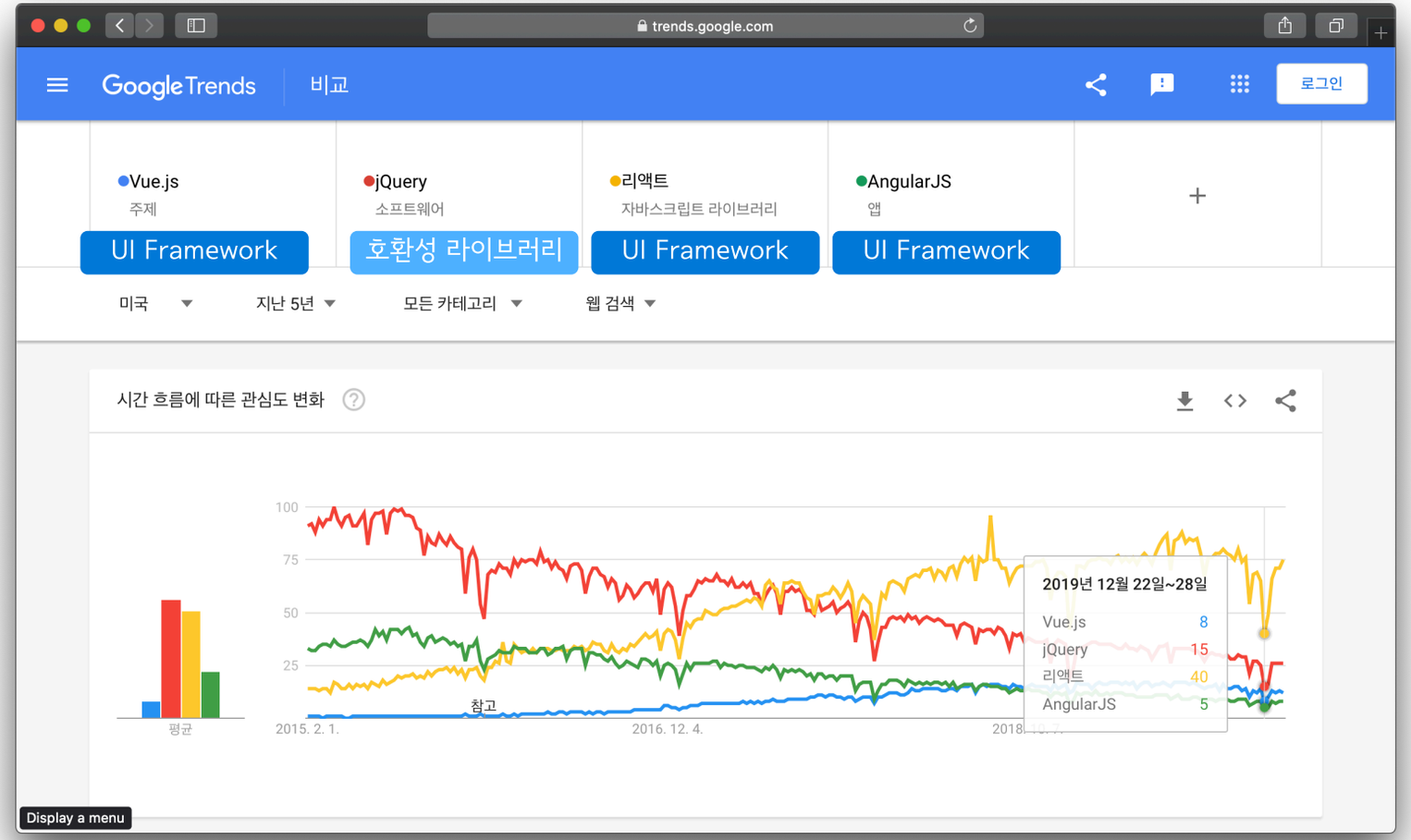
2006년, jQuery 출시

- 그 전에는 브라우저 별로 파편화가 무척 심했습니다. 프론트엔드 개발 HELL ~
- jQuery는 단일 API로 Ajax요청과 DOM조작을 편리하게 해줍니다.

하지만, 최근에는 jQuery를 안 쓰는 경우가 많음.

- 무거워진 jQuery, 하지만 방대한 기존 jQuery 플러그인
- 보다 근대화된 브라우저
- 보다 발전된 Web API
- 웹 페이지 개념에서는 먹히던 라이브러리 (웹 페이지의 시대 → 웹 애플리케이션의 시대)

Google Trends (2020년 2월 1일 기준)



https://trends.google.com/trends/explore?geo=US&q=vuejs,%2Fm%2F0268gyp,%2Fm%2F012l1vxv,%2Fm%2F0j45p7w

트렌드의 변화

구글검색: jQuery를 쓰면 안 되는 이유

대부분의 jQuery 메소드의 대안을 최근 브라우저에서는 네이티브 구현으로 제공

- 출처: You Don't Need jQuery

jQuery를 통해 먼저 프론트엔드를 접해보시는 것도 좋은 접근

우리는 jQuery를 통해, **장고 프로젝트를 좀 더 개선**해보도록 하겠습니다.

jQuery를 써봅시다.

django admin에서는 jQuery를 활용하고 있습니다.

jQuery CDN 버전 적용해보기

```
<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
<script>
// 이후에 필요한 자바스크립트 코드 구현
$(function() {
    console.log("웹페이지 로딩 완료");
});
</script>
```

브라우저의 개발자 도구 열기

프론트엔드 개발에서는 필히 브라우저의 "개발자 도구"를 열어주세요.

- 웹 개발 시의 브라우저는 **Chrome**이나 Firefox가 좋습니다.

브라우저 자바스크립트 수행 중에 발생한 오류는 "console 탭"에서 확인

구현해볼 기능

1. Event Listener 등록
   - 수많은 이벤트가 있지만, 이 중에 대표적으로 onload, click, submit 이벤트에 대해서만 살펴보겠습니다.
2. DOM 엘리먼트 추가/제거
3. Ajax GET/POST 요청

onload 이벤트 리스너 등록

이벤트 리스너 등록은 웹페이지 로딩 후에 하는 것이 안전합니다.

```
<script>
$(document).ready(function() {
    console.log("웹페이지 로딩 완료");
});
</script>
```

축약 표현

```
<script>
$(function() {
    console.log("웹페이지 로딩완료");
});
</script>
```

click 이벤트 리스너 등록

모든 DOM 엘리먼트에 등록 가능

```
<a id="btn-naver-1" href="http://m.naver.com" target="_blank">Naver Button 1</a>
<a id="btn-naver-2" href="http://m.naver.com" target="_blank">Naver Button 2</a>
<ul id="my-list">
    <li>list1</li>
    <li>list2</li>
    <li>list3</li>
</ul>
```

```
<script>
$(function() {
    // 리스너에 리턴값이 없기 때문에, 아래 리스너가 호출 + 본연의 href 링크가 동작
    $('#btn-naver-1').click(function() {
      console.log('clicked btn-naver-1');
    });

    // 아래 리스너가 호출되지만, href 링크는 동작하지 않습니다.
    $('#btn-naver-2').click(function(e) {
        e.preventDefault();  // 디폴트 동작 수행 방지. 혹은 return false; 도 동일한 효과
        console.log('clicked btn-naver-2');
        // return false;     // true를 리턴하면, 위 태그 클릭 시의 디폴트 동작 수행
    });

    $('#my-list li').click(function() {
        var content = $(this).html();
        console.log('clicked : ' + content);
    });
});
</script>
```

submit 이벤트 리스너 등록

```
<form id="query-form">
    <input type="text" name="query" />
    <input type="submit" value="조회" />
</form>

<script>
    $(function() {
        $('#query-form').submit(function(e) {
            e.preventDefault();
            console.log("form submit");
        });
    });
</script>
```

DOM 엘리먼트 추가/제거

```
<button id="lotto-btn">로또 번호를 점지해주세요.</button>
<button id="remove-at-first">처음을 삭제</button>
<button id="remove-at-last">마지막을 삭제</button>

<div id="lotto-list"></div>
```

```
<script>
$(function() {
    $('#lotto-btn').click(function() {
        var rendered = '<div class="post">로또 번호를 뽑아봅시다 : ' +
            (new Date()) + '</div>';

        // var container = $('#lotto-list').append(rendered);
        var $added = $(rendered).appendTo('#lotto-list'); // 추가된 jQuery객체를 리턴
        $added.click(function() {
            $(this).remove(); // 각 항목
        });
    });
    $('#remove-at-first').click(function() {
        $('#lotto-list div:first').remove();
    });
    $('#remove-at-last').click(function() {
        $('#lotto-list div:last').remove();
    });
});
</script>
```

jQuery에서 지원하는 Selector : https://api.jquery.com/category/selectors/

별도 템플릿을 통한 추가

복잡한 문자열을 custom script 태그를 통해 정의하기
Javascript로서 처리되지 않도록 임의 type 지정

```
<button id="lotto-btn">로또번호 생성하기</button>

<div id="lotto-list"></div>

<script id="post-template" type="text/x-template">
    <div class="post">
        당첨번호는 <%= numbers %>이며, 보너스 번호는 <%= bonus %>입니다.
    </div>
</script>

<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/underscore.js/1.9.1/underscore-min.js"></script>

<script>
$(function() {
    var raw_template = $('#post-template').html();
    var tpl = _.template(raw_template);

    $('#lotto-btn').click(function() {
        var sample = _.sample(_.range(1, 46), 6);

        var rendered = tpl({
            numbers: sample.slice(0, 5).sort(function(x, y) { return x - y; }),
            bonus: sample[5]
        });
        console.log(rendered);
        $(rendered).appendTo('#lotto-list');
    });
});
</script>
```

Ajax GET/POST 요청

HTTP 요청 Method

- GET 요청 : 주로 검색/조회/페이징처리 시에 사용
- POST 요청 : 주로 수정/삭제 처리 시에 사용

브라우저의 동일 도메인 참조 정책 (Same-Origin Policy)

- 같은 프로토콜/호스트명/포트 내에서만 Ajax 요청이 가능
- 초기에는 웹사이트의 보안을 위한 좋은 방법으로 생각되었으나, 최근 여러 도메인에 걸쳐서 구성되는 웹서비스가 늘어나는 시점에서는 거추장스러운(?) 기술??? 그래서, Cross Domain Request를 허용하기 위해 **CORS (Cross Origin Resource Sharing)** 지원 (서버측 셋업이 필요, 장고는 django-cors-headers 라이브러리 활용)
- 혹은 서버에 요청을 위임하여, 요청을 Proxy처리하기도 합니다. (ex: JSONP)

Django 뷰에서는 POST를 받을 때 CSRF Token값을 체크

- CSRF Token값이 없거나 값이 맞지 않으면 400 Bad Request 응답
- Django에서의 대응은 Django 공식문서를 참고

http://api.jquery.com/jquery.ajax/

CORS 설정: 모든 도메인에 대해서 허용한다면, 서버 측 응답헤더에 **Access-Control-Allow-Origin: *** 를 추가

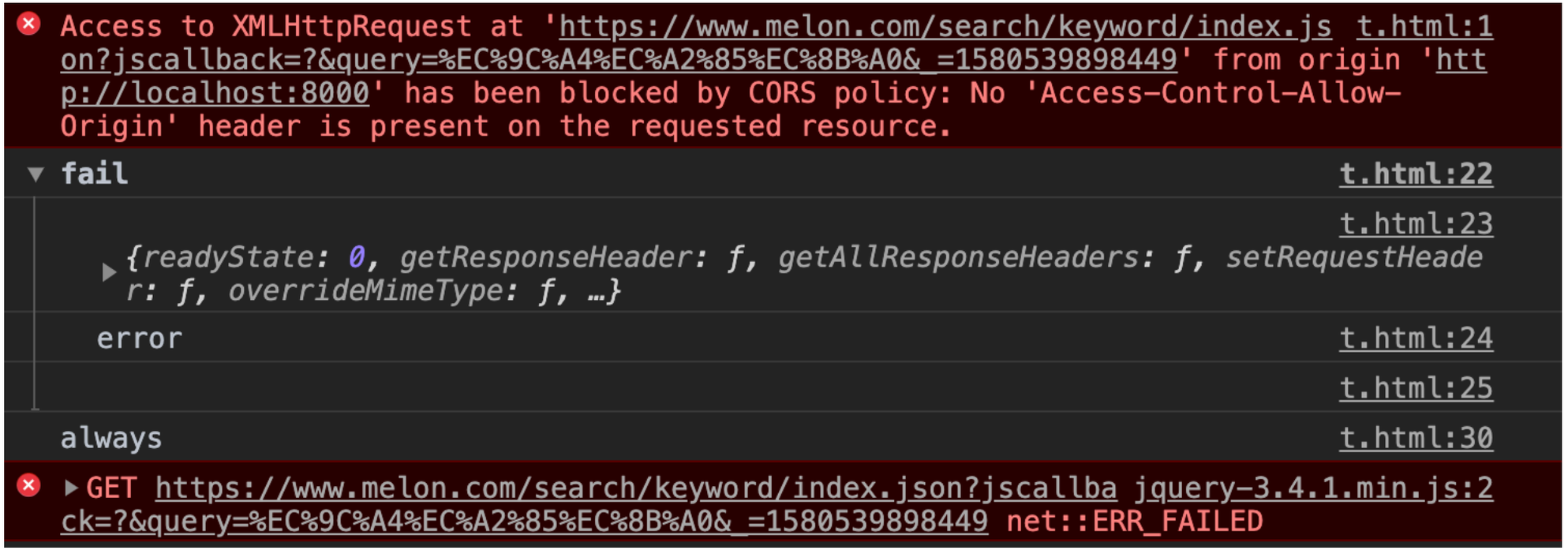
다른 도메인에 요청으로 하므로, 실패하는 Ajax GET 요청

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8" />
        <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
        <script>
            $.ajax({
                method: "GET",
                url: 'https://www.melon.com/search/keyword/index.json?jscallback=?',
                data: {query: '윤종신'},
                cache: false // true 지정 : GET인자로 "_={timestamp}" 를 붙여줍니다. 매번 URL이 달라지므로 브라우저 캐싱이 되지 않습니다.
            })
            .done(function(response, textStatus, xhr) {
                console.group("done");
                console.log(response)
                console.log(textStatus);
                console.log(xhr);
                console.groupEnd();
            })
            .fail(function(xhr, textStatus, error) {
                console.group("fail");
                console.log(xhr)
                console.log(textStatus);
                console.log(error);
                console.groupEnd();
            })
            .always(function(response_or_xhr, textStatus, xhr_or_error) {
                // 인자가 done/fail 인자 매핑과 동일
                console.log("always");
            });
        </script>
    </head>
</html>
```

다른 도메인

같은 도메인이라면 성공

이전 요청에 대한, 개발자도구에서의 오류 메세지

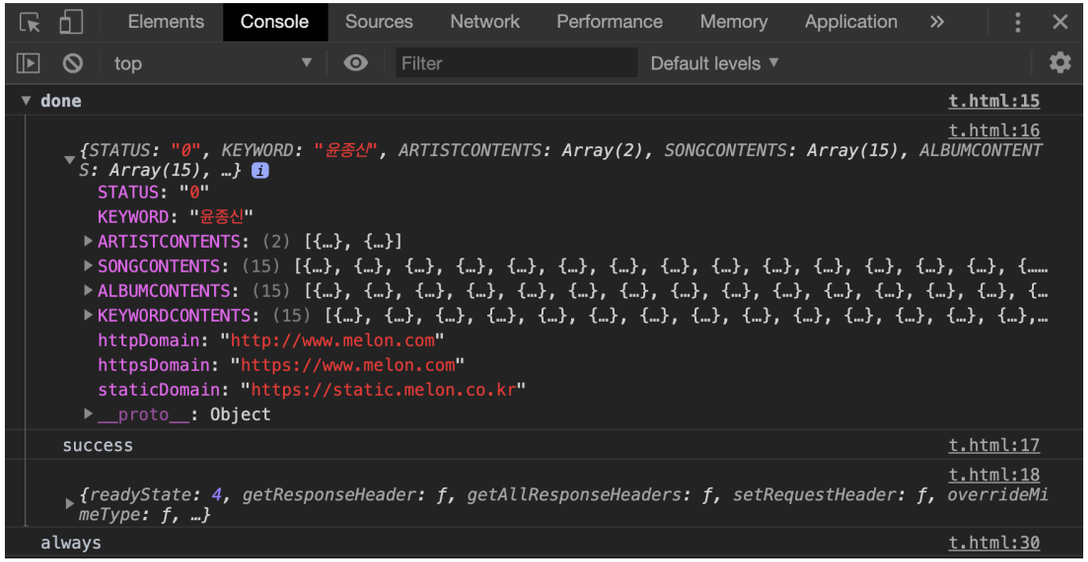


교차 원본 요청 차단: 동일 출처 정책으로 인해 [요청한 도메인]에 있는 원격 자원을 읽을 수 없습니다. 자원을 같은 도메인으로 이동시키거나 CORS를 활성화하여 해결할 수 있습니다.

JSONP의 예

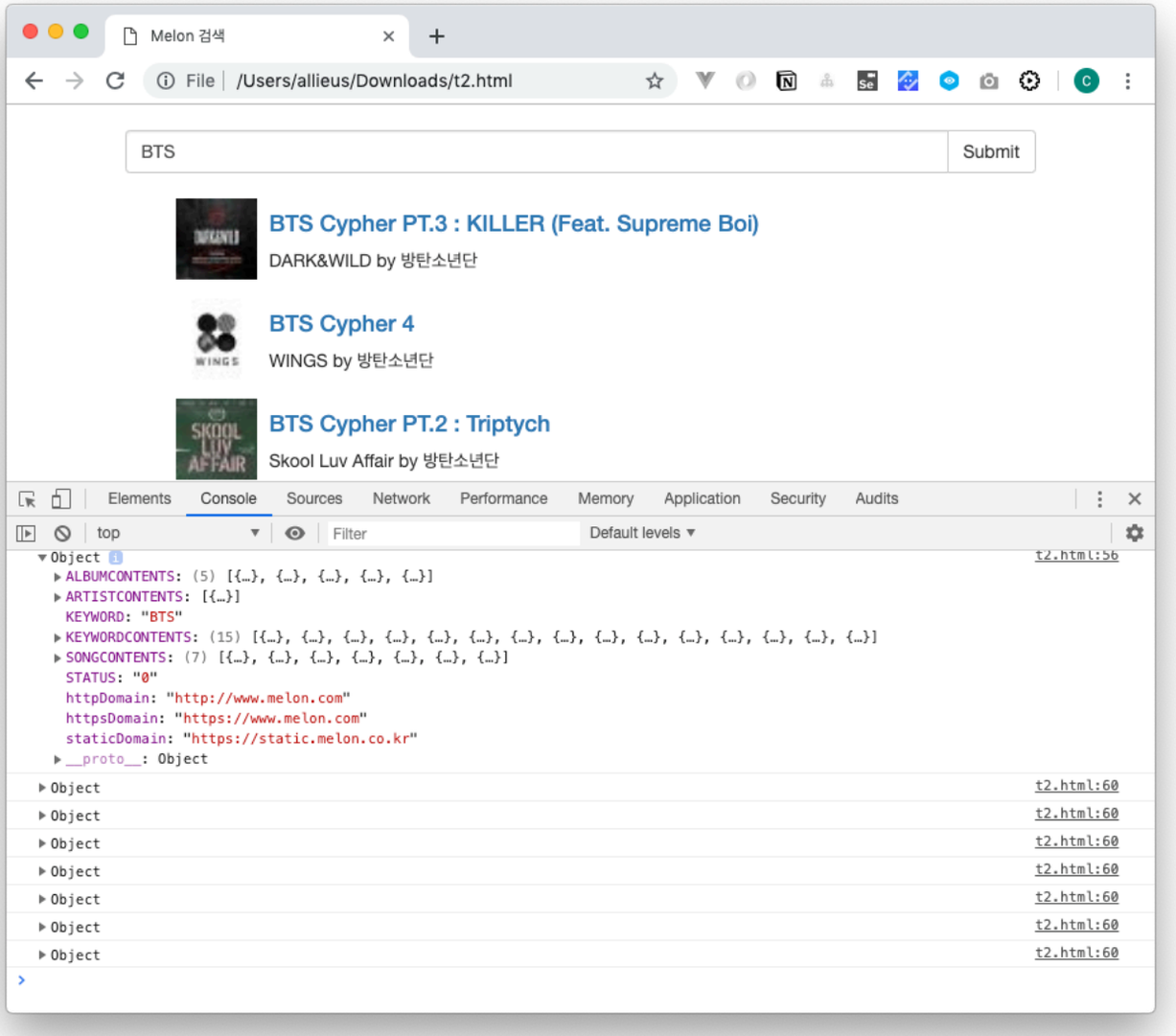
```html
<!doctype html>
<html>
    <head>
        <meta charset="utf-8" />
        <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
        <script>
            $.ajax({
                method: "GET",
                url: 'https://www.melon.com/search/keyword/index.json?jscallback=?',
                data: {query: '윤종신'},
                dataType: 'jsonp',
                cache: false // true 지정 : GET인자로 "_={timestamp}" 를 붙여줍니다. 매번 URL이 달라지므로 브라우저 캐싱이 되지 않습니다.
            })
            .done(function(response, textStatus, xhr) {
                console.group("done");
                console.log(response)
                console.log(textStatus);
                console.log(xhr);
                console.groupEnd();
            })
            .fail(function(xhr, textStatus, error) {
                console.group("fail");
                console.log(xhr)
                console.log(textStatus);
                console.log(error);
                console.groupEnd();
            })
            .always(function(response_or_xhr, textStatus, xhr_or_error) {
                // 인자가 done/fail 인자 매핑과 동일
                console.log("always");
            });
        </script>
    </head>
</html>
```

JSONP: `<script />` 태그는 SOP정책에 속하지 않음을 이용하여, 서로 다른 도메인/포트 간에 JSON GET통신을 할 수 있는 방법



실습 : 멜론으로부터 검색결과 받아서 보여주기

https://goo.gl/JisDtS



Life is short.
You need Python and Django.

I will be your pacemaker.