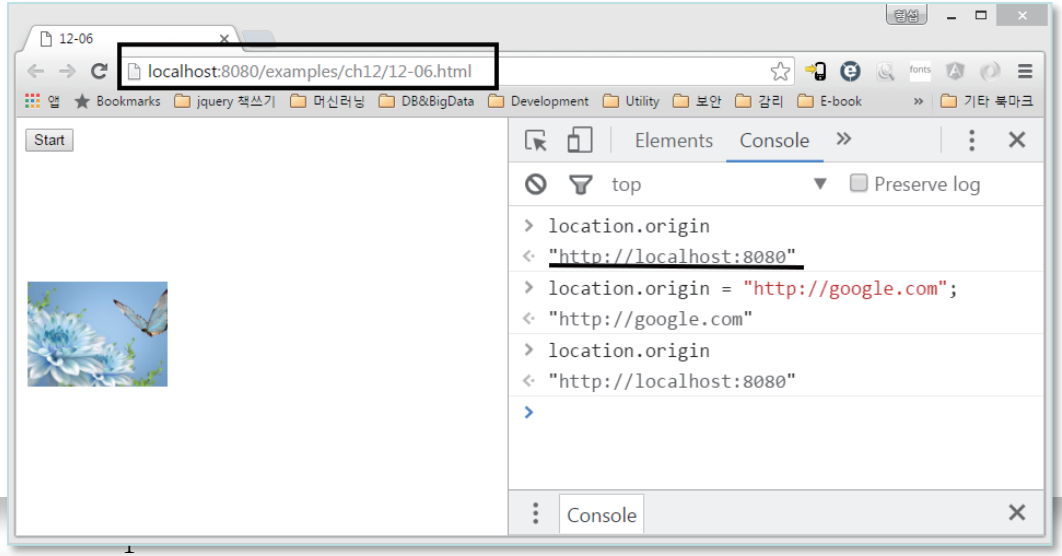
크로스 도메인 문제란?(1)

- 정의
  - "동일 근원 정책(SOP : Same Origin Policy)이라는 웹브라우저의 보안 정책으로 인해, 현재 브라우저와 동일한 오리진(Origin)이 아닌 다른 오리진으로부터 AJAX를 이용해 데이터를 로드하지 못한다."
  - 정확하게 개념을 이해하려면 오리진(Origin)이 무엇인지부터 이해해야 함.
- 오리진?
  - 브라우저에서 현재 보여지고 HTML 문서가 어디에서 가져온 것인지를 나타내는 정보. 브라우저의 콘솔에서 직접 확인 가능
  - 오리진은 직접 수정 불가

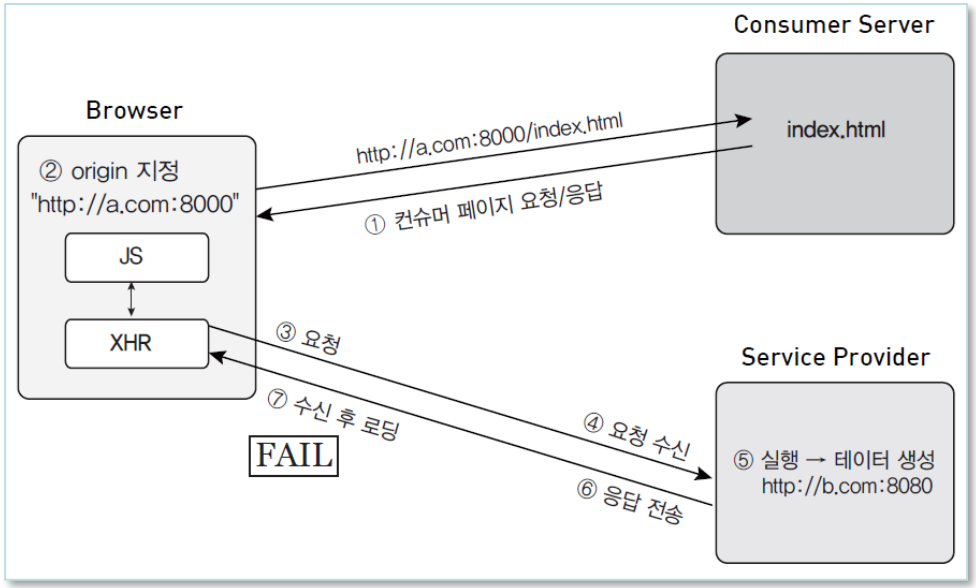


크로스 도메인 문제란?(2)

- (이어서)
- Origin은 Domain Name이 아님
  - "http://localhost:8080"
  - 리소스 + 주소 + 포트번호
  - 단순 문자열임. 이 문자열이 다르면 다른 오리진
  - 동일한 장비를 사용하더라도 다른 주소나 이름을 사용하면 다른 오리진으로 구분
  - 오리진만 이해했다면 크로스 도메인 문제는 의외로 간단함.

크로스 도메인 문제란?(3)

크로스 도메인 문제 개념도



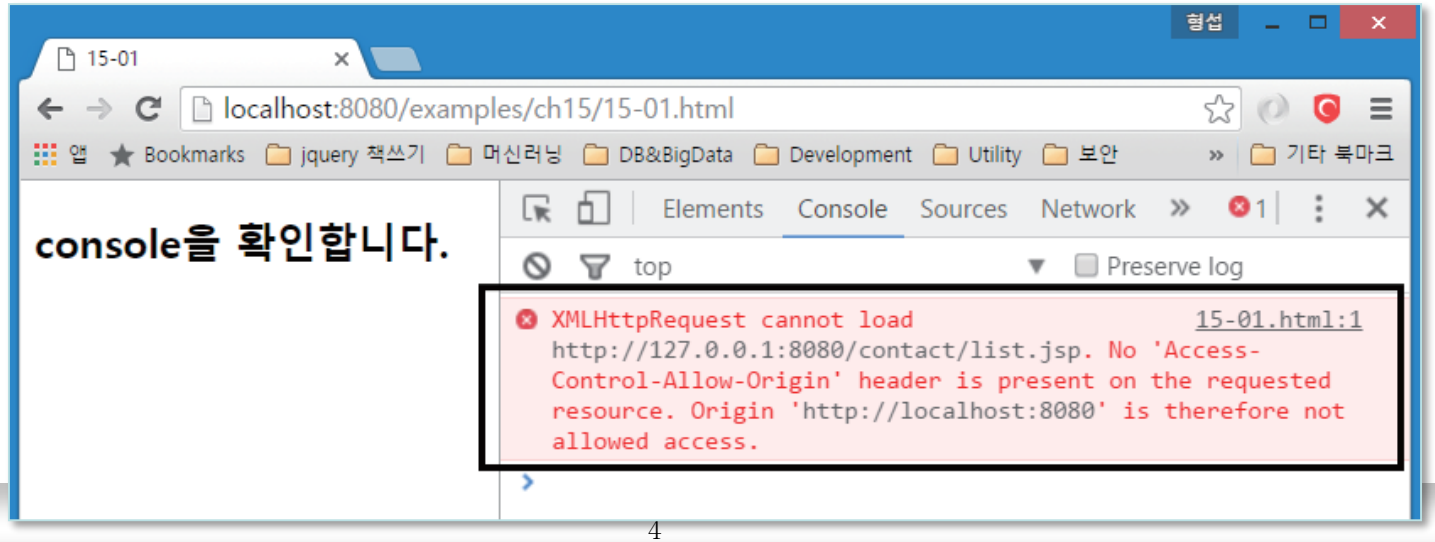
- 크로스 오리진 상황이라도 요청~응답전송까지는 정상 수행
- 마지막 단계에 브라우저가 수신 데이터를 로드하지 않음
  - 통신할 수 없다(X) 통신은 하지만 데이터를 로드하지 않음(O)

크로스 도메인 문제란?(4)

예제 15-01 : 크로스 도메인 문제 발생

[예제 15-01]

```
01: <script src="http://code.jquery.com/jquery-3.1.0.js"></script>
02: <script type="text/javascript">
03: $(document).ready(function() {
04:     var url = "http://127.0.0.1:8080/contact/list.jsp";
05:     $.get(url, function(data) {
06:         console.log(data);
07:     });
08: });
09: </script>
```



크로스 도메인 문제란?(5)

- 크로스 도메인 상황에서 데이터 추가/수정 작업
  - 추가/수정 작업 결과 데이터를 로드하지 못한다.
  - 추가/수정 작업이 수행된다.

동일 근원 정책(SOP:Same Origin Policy)는 왜?

- 컨슈머로부터 내려받은 자바스크립트 코드가 사용자가 알지 못한 상태에서 다른 오리진으로부터 출처를 알 수 없는 데이터나 악성 코드를 내려받지 못하게 막기 위한 목적으로 적용된 것
- 하지만 지금은 크로스 도메인 간의 데이터 요청/응답이 꼭 필요함
- 거추장스러운 보안 정책!
- 이 문제를 극복할 수 있는 방법이 생겨났음

크로스 도메인 문제 해결 방법(1)

크로스 도메인 문제를 해결

- 문제 해결을 위해 브라우저의 옵션을 설정할 수 있지만 모든 사용자에게 설정을 강요할 수는 없음
- 브라우저의 설정을 변경하지 않고도 해결할 수 있는 방법이 필요

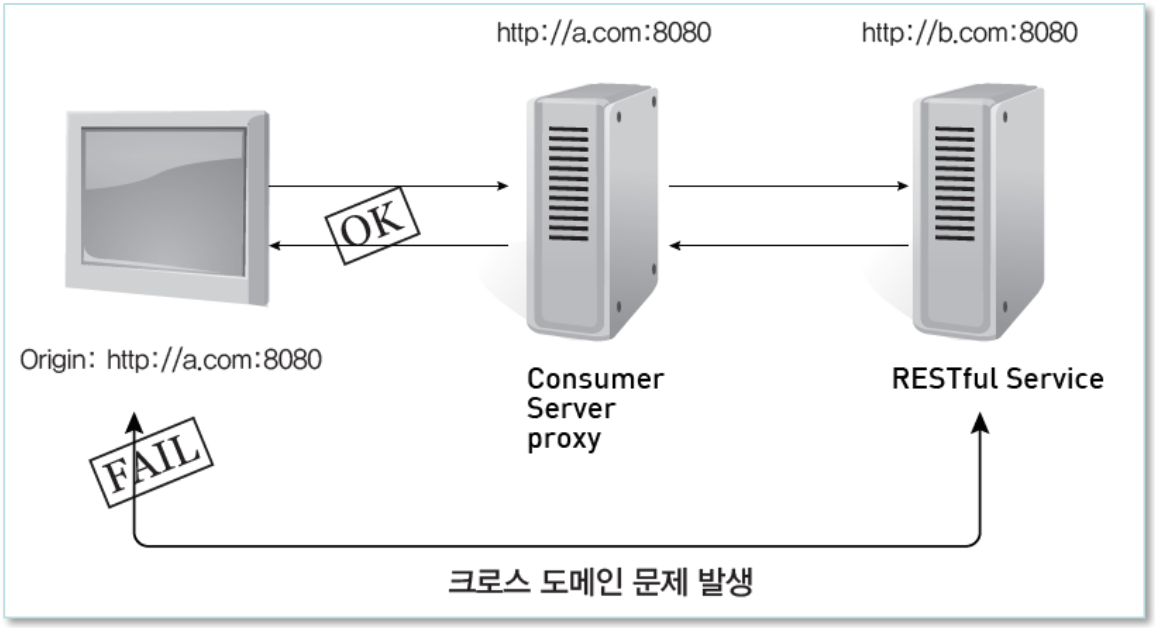
크로스 도메인 문제 해결 방법

- 컨슈머측 해결 방법
  - 컨슈머 서버 프록시
- 프로바이더측 해결 방법
  - CORS(Cross Origin Resource Sharing)
  - JSONP(JSON Padding)

크로스 도메인 문제 해결 방법(2)

컨슈머 서버 프록시

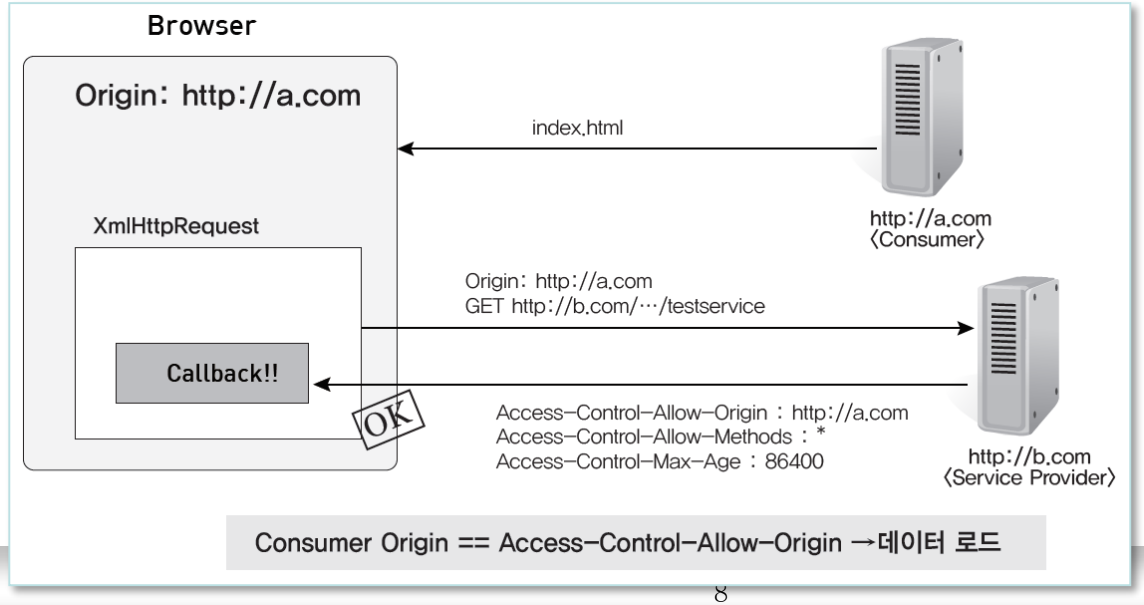
- 서비스 프로바이더측에서 아무런 방법을 제공하지 않을 때 사용
  - 브라우저가 요청하는 경로를 컨슈머 서버의 프록시를 향하도록 하고 프록시가 서비스 쪽으로 대신 요청하고 응답 데이터가 수신되면 브라우저로 전달해주는 방법
  - 컨슈머 앱 개발자가 직접 프록시 요소를 작성해야 하는 불편함이 있음.



크로스 도메인 문제 해결 방법(3)

CORS(Cross Origin Resource Sharing)

- "서비스 제공자(Service Provider)가 허락하는 경우에 브라우저가 데이터를 로드할 수 있도록 한다."
  - 서비스 제공자가 Access-Control-Allow-Origin 헤더로 브라우저의 오리진을 지정해 응답하면 브라우저가 거부하지 않고 데이터를 로드할 수 있음
  - 서비스 제공자는 브라우저가 전송해온 Origin 헤더, Referer 헤더를 확인해 컨슈머를 확인하는 것이 바람직하다.



크로스 도메인 문제 해결 방법(4)

CORS(이어서)

- 브라우저는 자신의 오리진과 Access-Control-Allow-Origin 헤더값을 비교해 일치하면 데이터를 로드함
- Access-Control로 시작하는 헤더를 이용해 추가적인 정보를 제공할 수 있음
  - Access-Control-Allow-Methods : 이 값을 GET으로 지정하면 GET 방식만 지원
- 연락처 서비스 프로젝트에 list_cors.jsp 추가
  - list.jsp를 복사한 후 다음 코드만 추가

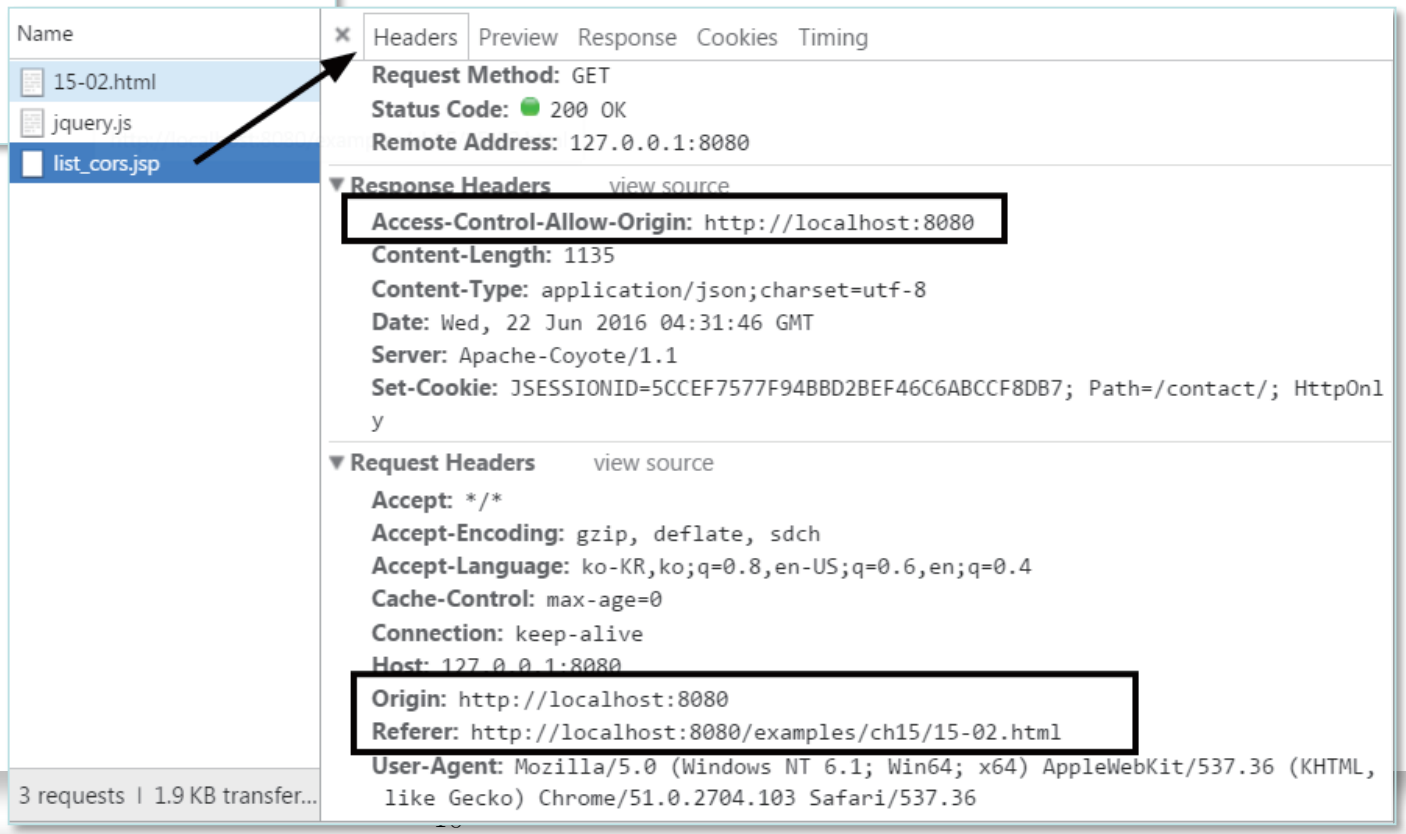
```
27:     //사전에 등록된 오리진은 http://localhost:8080
28:     String origin = request.getHeader("origin");
29:     if (origin.equals("http://localhost:8080")) {
30:         response.setHeader("Access-Control-Allow-Origin", origin);
31:     }
```

크로스 도메인 문제 해결 방법(5)

- 예제 15-02를 복사한 후 요청 URL 만 변경한 후 실행

[예제 15-03 : CORS 서비스 호출]

```
01: <script src="http://code.jquery.com/jquery-3.1.0.js"></script>
02: <script type="text/javascript">
03: $(document).ready(function() {
04:     var url = "http://127.0.0.1:8080/contact/list_cors.jsp";
05:     $.get(url, function(data) {
06:         console.log(data);
07:     });
08: });
09: </script>
```



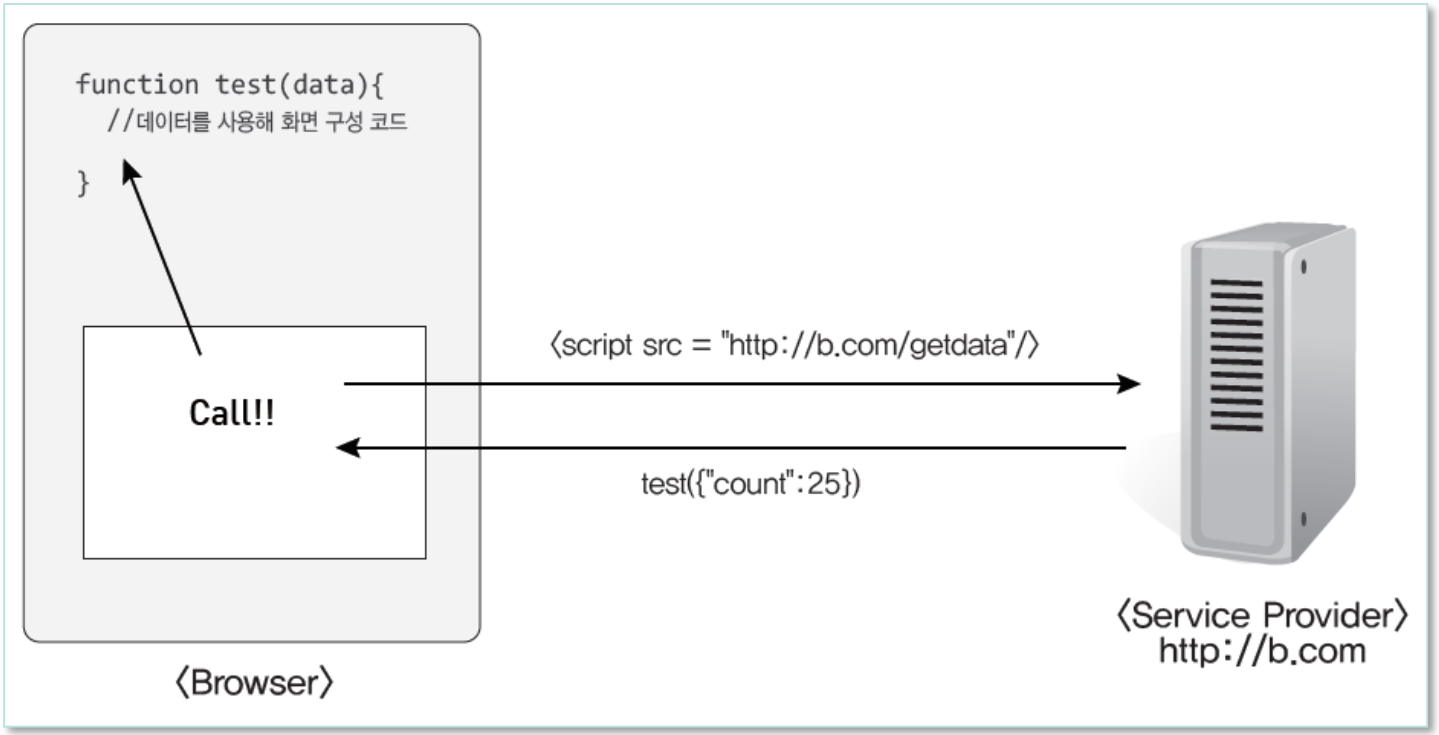
크로스 도메인 문제 해결 방법(6)

- 중요한 점은 서비스 제공자가 CORS 기법을 제공하면 AJAX 관련 코드는 서비스의 주소 URL을 제외하면 변경할 것이 거의 없다는 것이다.
- CORS도 자세하게 들어가면 단순 요청(Simple Request), 예비 요청(Preflight Request), 인증 요청(Request with Credential) 방식 등 상세한 방법이 있지만 이 책에서는 개념수준에서만 살펴본다

크로스 도메인 문제 해결 방법(7)

JSONP(JSON Padding)

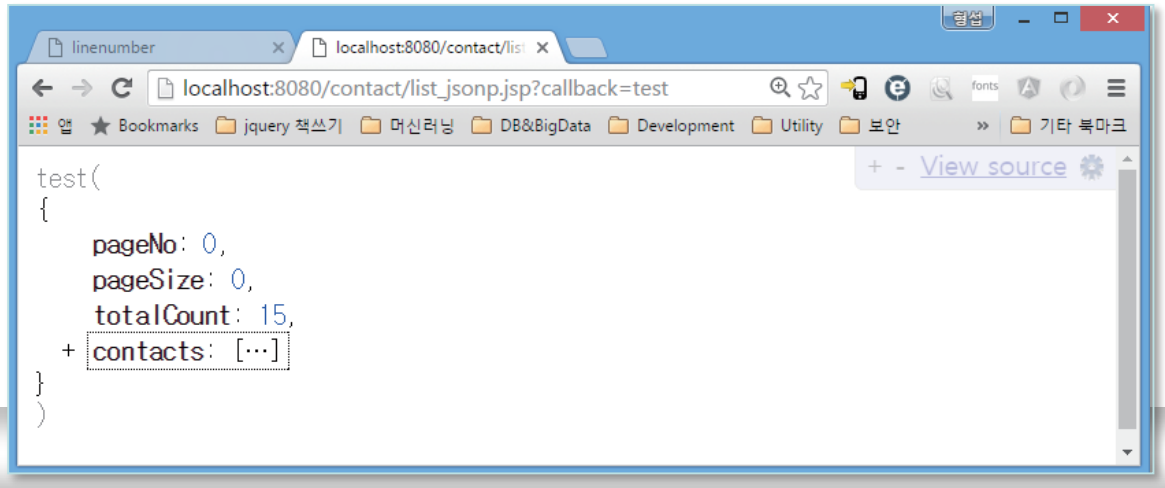
- 자바스크립트 파일을 로드하기 위해 사용하는 <script>태그의 src 속성은 SOP(Same Origin Policy)의 영향을 받지 않는다는 점을 활용한 기법



크로스 도메인 문제 해결 방법(8)

- <script> 태그의 src 특성값에 서비스 제공자의 주소를 입력
- 서비스 제공자는 test({ … }) 형식으로 응답
  - 순수한 JSON 문자열 앞과 뒤에 "test(", ")" 을 Padding함.
  - test와 같은 호출 함수명은 지정 가능하도록 작성해야 함.
- 예제 15-04 : list_jsonp.jsp

```
28:     String callback = request.getParameter("callback");
29:     if (callback != null && !callback.equals("")) {
30:         json = callback + "(" + json + ")";
31:     }
32: %><%=json %>
```



크로스 도메인 문제 해결 방법(9)

- JSONP 서비스 호출 원리

[예제 15-05]

```
<script type="text/javascript">
function test(data) {
   console.log(data);
}
</script>
<script src="http://127.0.0.1:8080/contact/list_jsonp.jsp?callback=test&pageno=2"></script>
```

[예제 15-06]

```
<script src="http://code.jquery.com/jquery-3.1.0.js"></script>
<script type="text/javascript">
function test(data) {
   console.log(data);
}

$(document).ready(function() {
   $.getScript("http://127.0.0.1:8080/contact/list_jsonp.jsp?callback=test&pageno=2");
});
</script>
```

크로스 도메인 문제 해결 방법(10)

- jQuery AJAX의 JSONP 지원 기능
  - AJAX 호출은 아니다. AJAX 호출과 형식은 같지만 스크립트 태그를 동적으로 추가하는 방법을 사용한다.
  - $.ajax(), $.getJSON()

```
09:     var url1 = "http://127.0.0.1:8080/contact/list_jsonp.jsp";
10:     $.ajax({
11:         url : url1,
12:         type : "GET",
13:         data : { pageno:1, pagesize:3 },
14:         dataType : "jsonp",
15:         jsonp : "callback",
16:         success : function(data) {
17:             console.log(data);
18:         }
19:     });
20:
21:     var url2 = "http://127.0.0.1:8080/contact/list_jsonp.jsp?callback=?"
22:     $.getJSON(url2, { pageno:2, pagesize:3 }, function(data) {
23:         console.log(data)
24:     });
```

크로스 도메인 문제 해결 방법(11)

- $.ajax()
  - dataType 옵션을 jsonp로 지정
  - jsonp 옵션을 callback으로 지정. callback 파라미터 명을 지정한다.
  - success 콜백 함수를 지정하여 응답결과 수신
- $.getJSON()
  - "?callback=?"과 같이 두번째 ? 기호를 사용함.
- 호출 결과