12

HTTP와 쿠키, 웹 스토리지

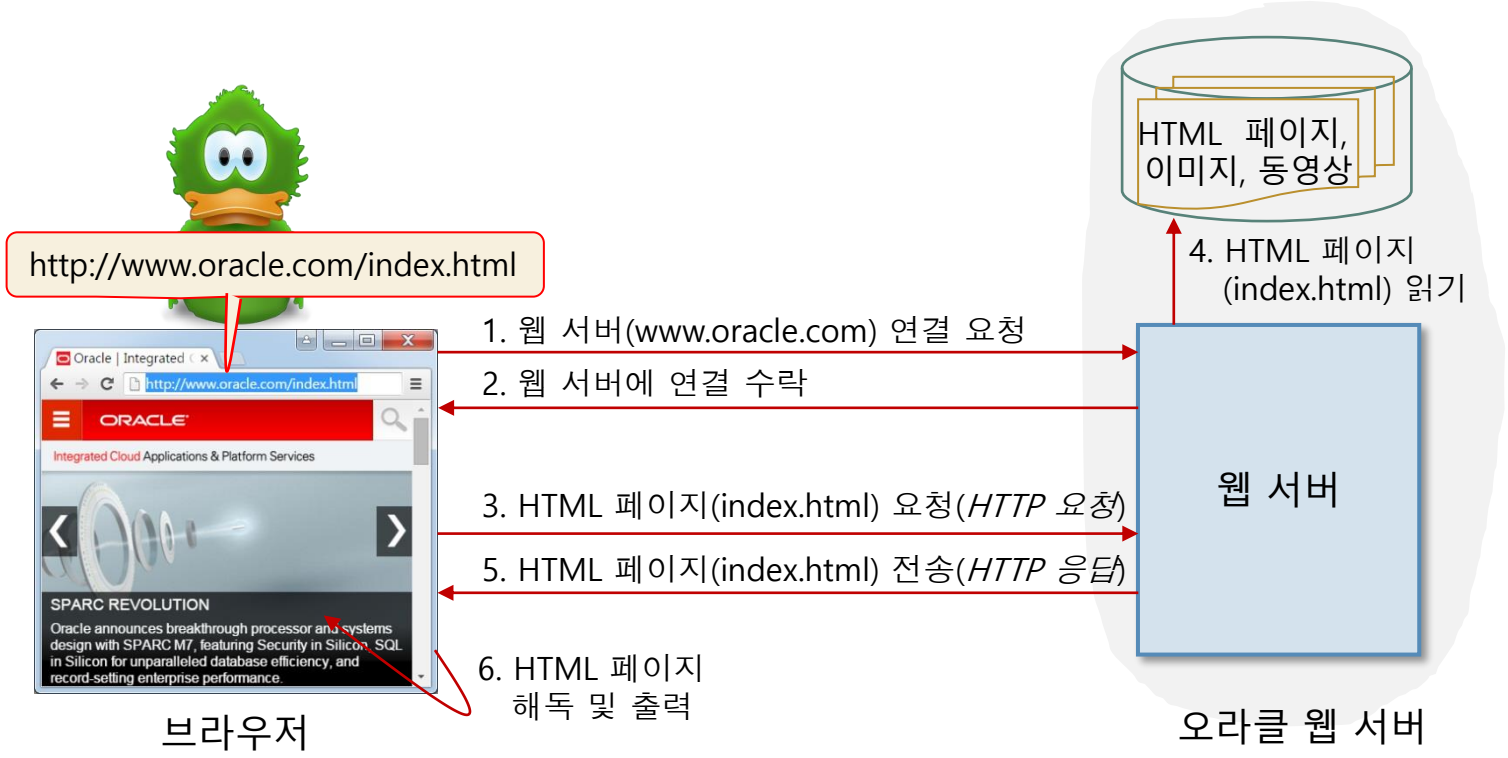
강의 목표

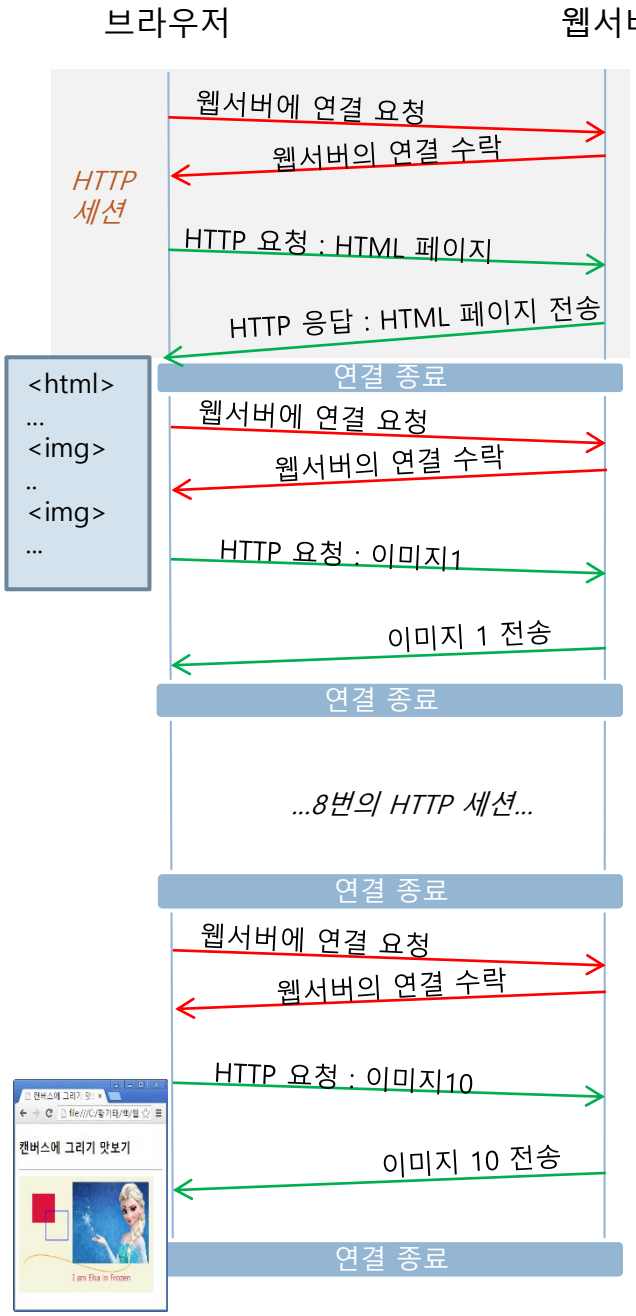
1. 브라우저와 웹 서버 사이의 통신(HTTP) 과정을 이해한다.
2. 실습을 통해 브라우저와 웹 서버 사이의 통신(HTTP) 과정을 확인한다.
3. 쿠키 데이터의 목적과 누가 생산하고 누가 저장하며, 어디에 저장되는지 안다.
4. 자바스크립트 코드로 쿠키를 만들고 읽을 수 있다.
5. 웹 스토리지(세션 스토리지, 로컬 스토리지)를 자세히 이해한다.
6. 자바스크립트로 세션 스토리지와 로컬 스토리지에 값을 저장하고 읽을 수 있다.
7. 웹 스토리지를 응용할 수 있다.

웹의 저장소

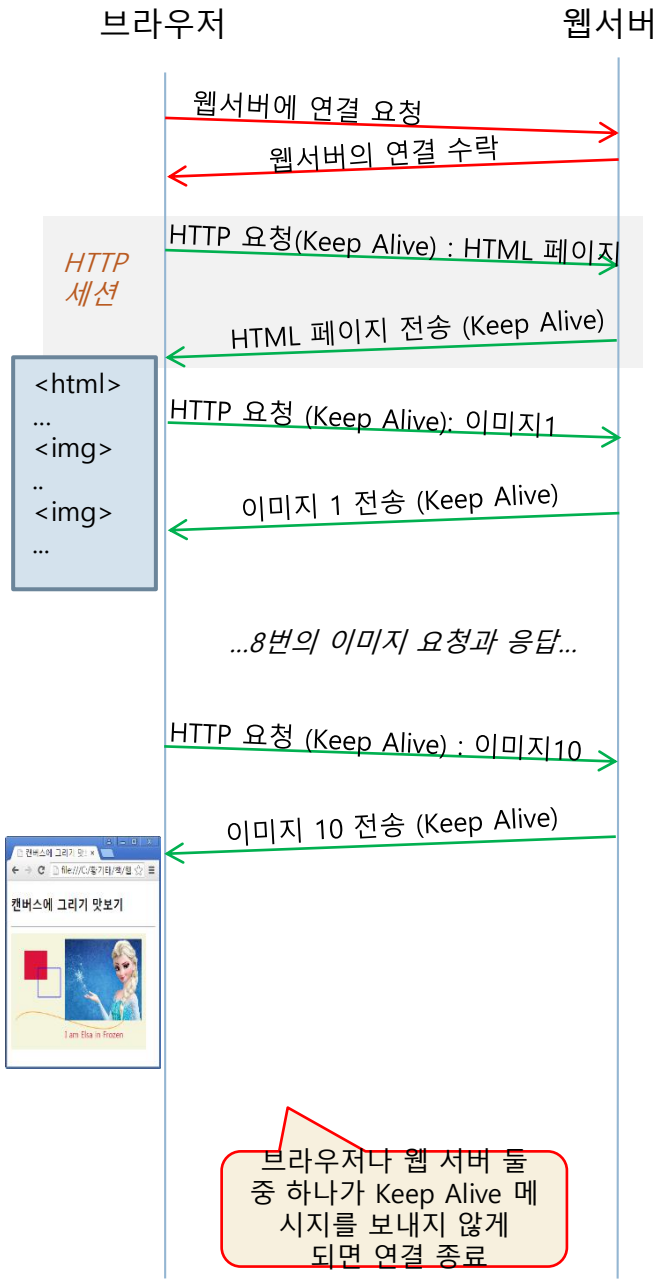
- 초기 웹의 저장소
  - 웹 서버에 저장
    - HTML 페이지, 이미지, 사용자 데이터, 웹 서비스 중간에 발생하는 일시적인 데이터
  - 초기 웹 저장의 문제점
    - 웹 사용의 폭발적 증가
    - 웹 서버의 저장 용량에 대한 부담
    - 웹 브라우저와 웹 서버 사이의 통신 트래픽 증가
- HTML5의 웹 저장소
  - 사용자의 로컬 컴퓨터에 일부 데이터 저장
    - 웹 서버의 저장 용량 및 통신 트래픽 감소
  - 저장소 종류
    - 쿠키(Cookie)
    - 웹 스토리지(Web Storage)
    - 로컬 파일(Local File)
    - 인덱스트 데이터베이스(Indexed DB)
  - 웹 서버와 연결이 끊어진 경우에도 로컬 컴퓨터에도 웹 애플리케이션 실행

브라우저와 웹 서버의 통신, HTTP

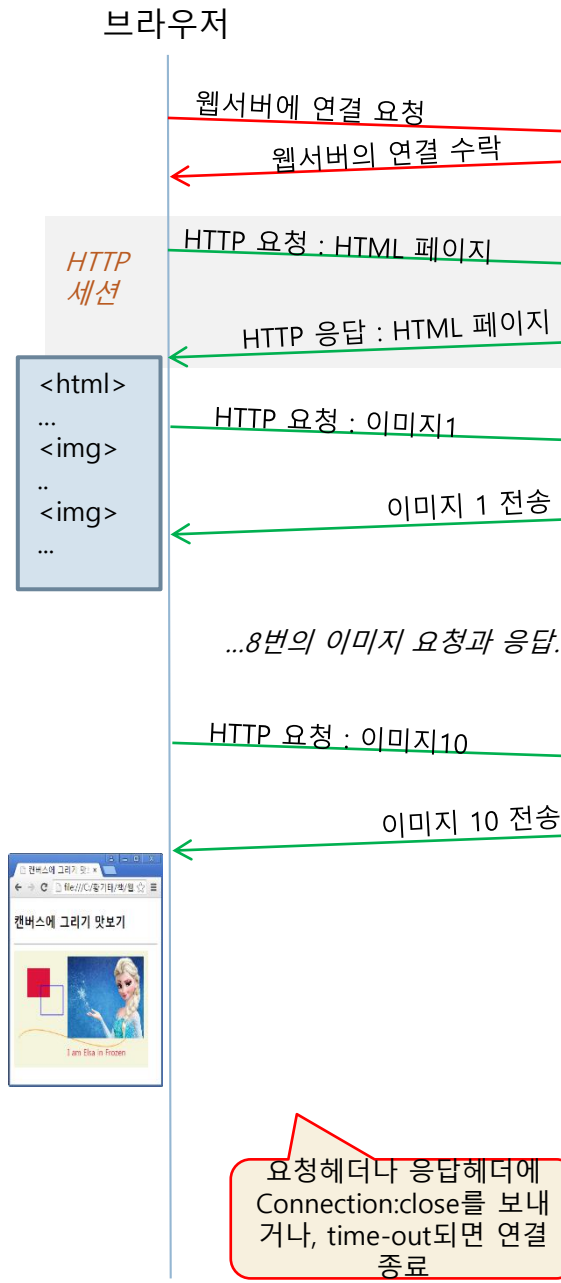




(a) HTTP/1.0
Connectionless Protocol



(b) HTTP/1.0, HTTP/1.1
Keep Alive

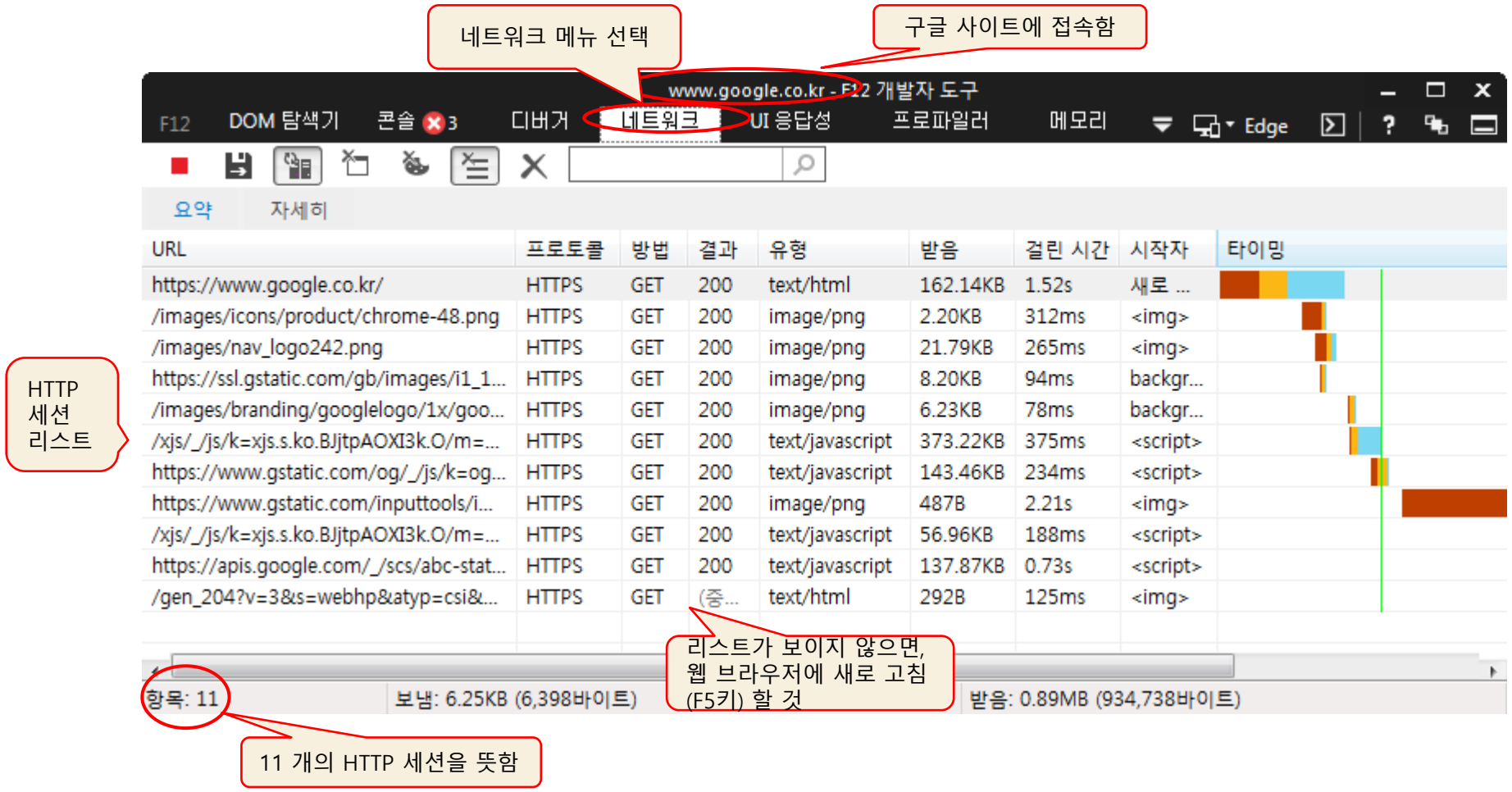


(c) HTTP/1.1
Persistent Connection

실습 1 : HTTP 통신 과정 보기

- 인터넷 익스플로러의 개발자 도구 열기 – F12 키

인터넷 익스플로러의 개발자 도구를 별도 윈도우에서 실행

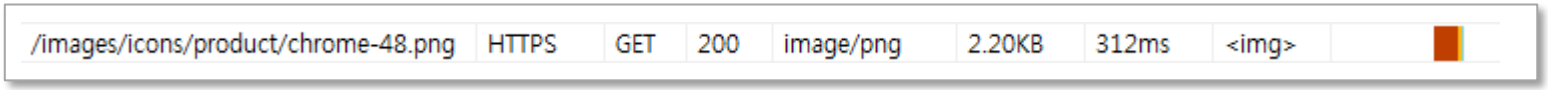


HTTP 통신 과정 보기

- 첫 번째 HTTP 세션

  - 익스플로러가 www.google.co.kr 웹 서버에 연결
  - 디폴트 HTML 파일을 요청하고 응답 받았음
  - 총 1.52초 걸렸다는 뜻

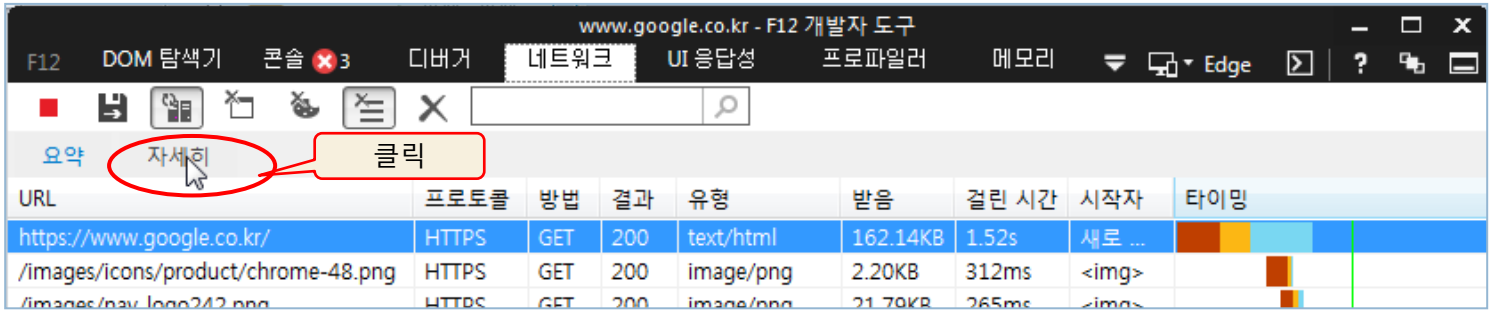
- 두 번째 HTTP 세션



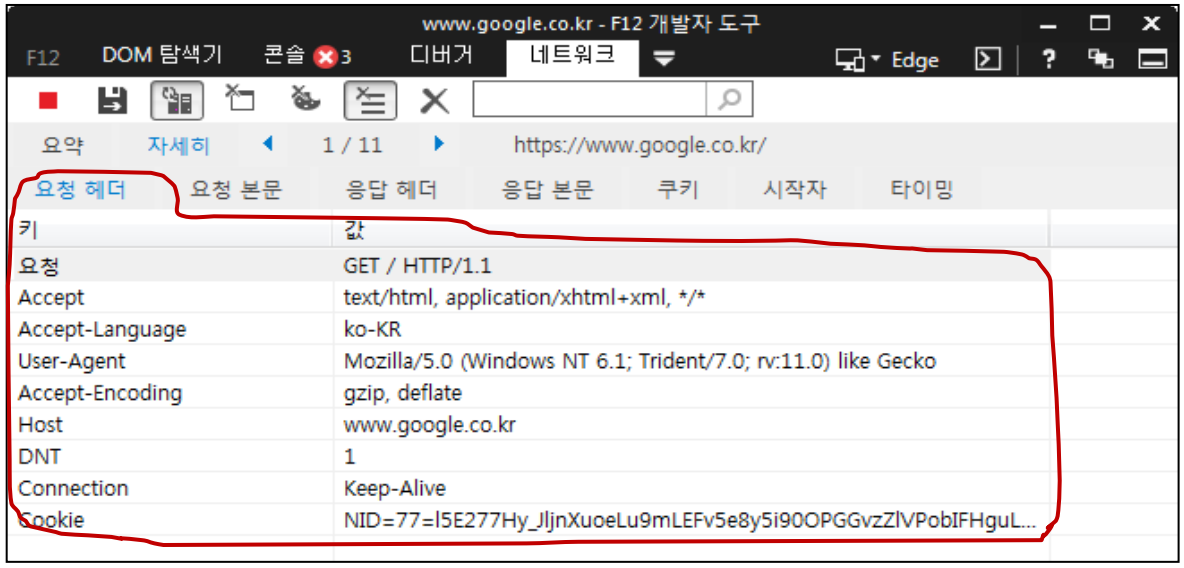
  - <img>의 src 속성에 명시된 chrome-48.png을 구글 웹 서버에 요청
  - 요청을 포함하여 이미지를 전송 받는데 거리는 시간 총 312ms
  - 이미지의 크기는 2.20K

HTTP 요청 헤더 보기

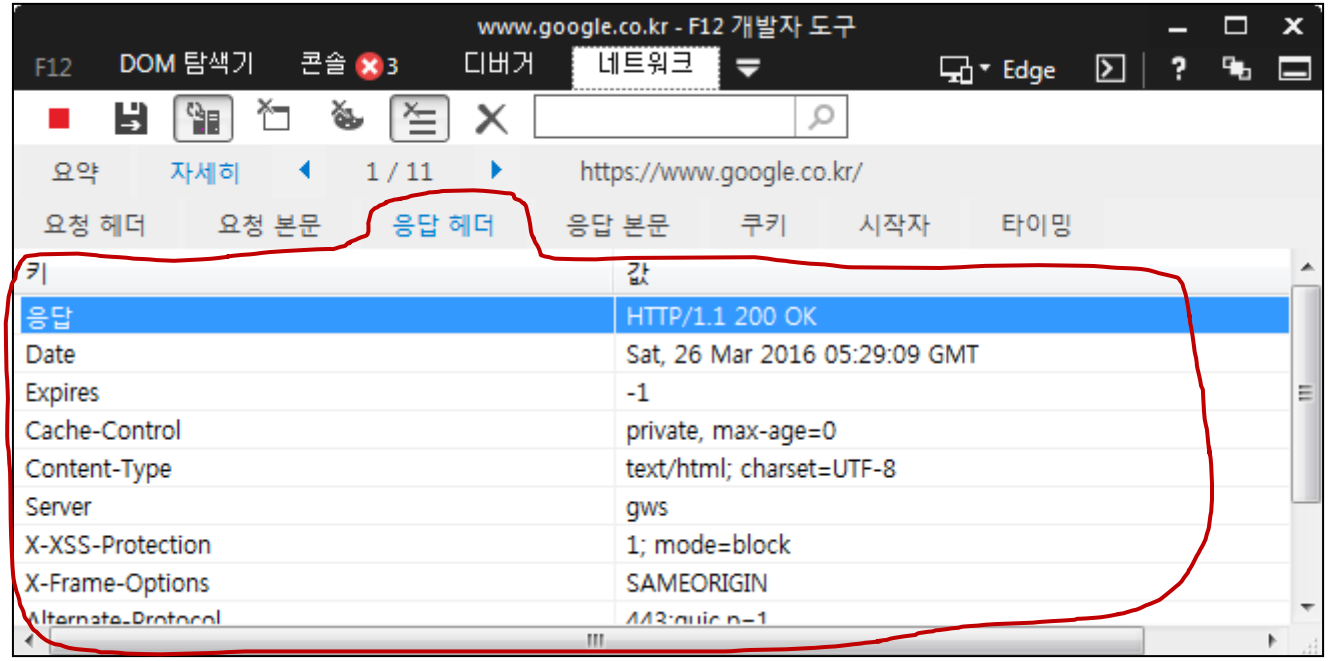
- 첫번째 HTTP 세션에 대한 자세한 정보 보기



- 첫 번째 HTTP 세션의 HTTP 요청 헤더



HTTP 응답 헤더

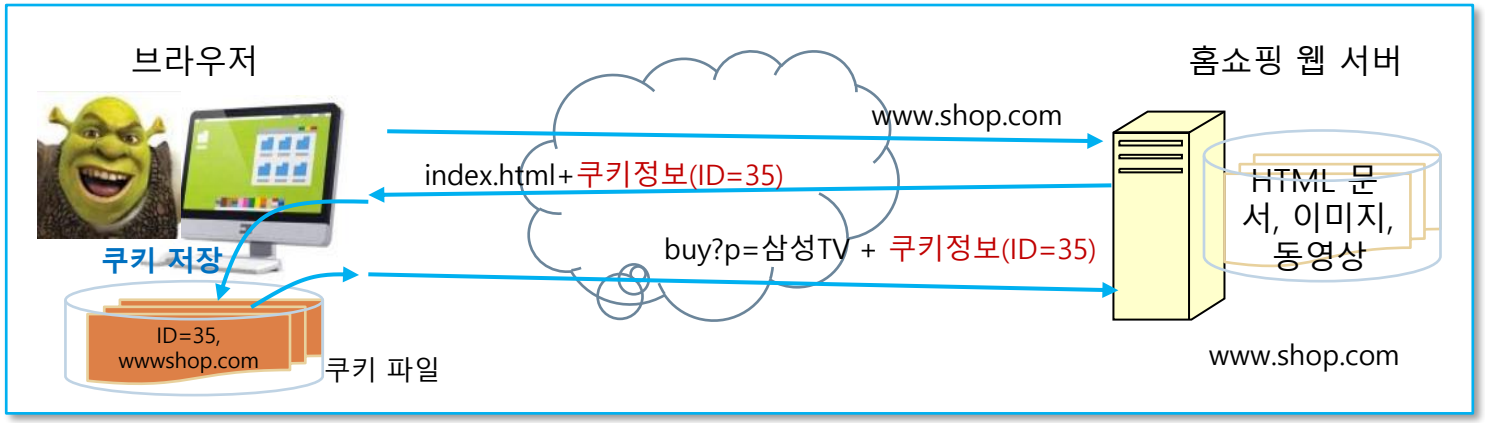

www.google.co.kr - F12 개발자 도구
F12
DOM 탐색기
콘솔
3
디버거
네트워크
Edge
요약
자세히
1 / 11
https://www.google.co.kr/
요청 헤더
요청 본문
응답 헤더
응답 본문
쿠키
시작자
타이밍
키
값
응답
HTTP/1.1 200 OK
Date
Sat, 26 Mar 2016 05:29:09 GMT
Expires
-1
Cache-Control
private, max-age=0
Content-Type
text/html; charset=UTF-8
Server
gws
X-XSS-Protection
1; mode=block
X-Frame-Options
SAMEORIGIN

쿠키

- 쿠키란?
  - 웹 서버가 브라우저에게 지시하여 사용자 로컬 컴퓨터에 저장하는 4K 이하의 작은 데이터
- 쿠키의 도입 배경
  - HTTP의 통신의 기본 약점
    - 브라우저와 웹서버 사이의 통신은 무상태(stateless) 프로토콜임
    - 무상태 프로토콜
      - 바로 이전 요청과 현재 요청이 연결되어 있음을 기억하지 않는 통신
    - 예) 지금'Java'를 검색하는 사용자가 바로 전에 'C++'를 검색한 사용자라는 사실을 모른다
  - 쿠키는 HTTP 의 무상태 프로토콜의 약점을 보완하기 위해 도입

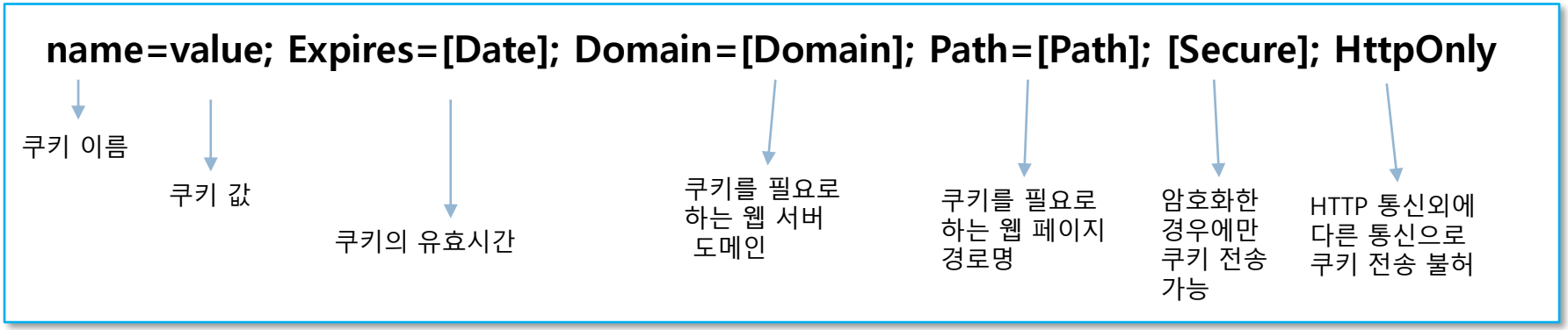
쿠키 생성 및 사용 과정

- 1. 쿠키는 웹 서버가 생성하여 브라우저로 보냄
  - 사용자가 어떤 웹 서버에 처음 접속할 때
  - 웹 서버가 다음 요청에서 그 사용자를 기억할 수 있도록 쿠키(쿠키이름과 값)를 만들어 전송
- 2. 쿠키를 받은 브라우저는 로컬 컴퓨터에 저장
- 3. 로컬 컴퓨터에서 동일한 웹 서버에 요청할 때 쿠키를 함께 전송
  - 웹 서버로 요청하는 경우 : 웹 페이지 요청, 이미지 요청 등 모든 웹 자원 요청 포함
- 4. 쿠키를 받은 웹 서버는 어떤 사용자로부터 요청이 왔는지 알 수 있음



쿠키 데이터 구성과 사례

- 쿠키 구성 : 6 개의 속성으로 구성

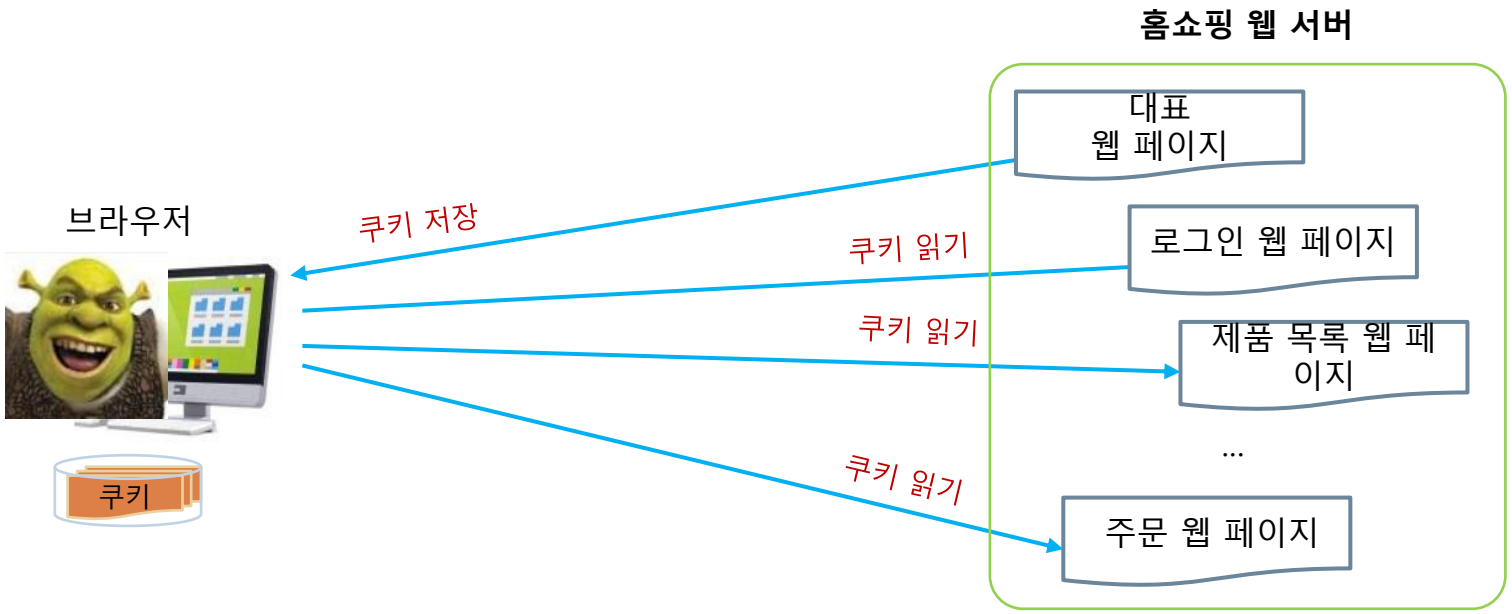


  - 브라우저가 웹 서버로 쿠키를 전송할 때는 name=value만 전송
- 쿠키 사례

```
age=23; expires=Mon, 01-Aug-2016 00:00:01 GMT; Domain=.google.com; Path=/; Secure; HttpOnly
```

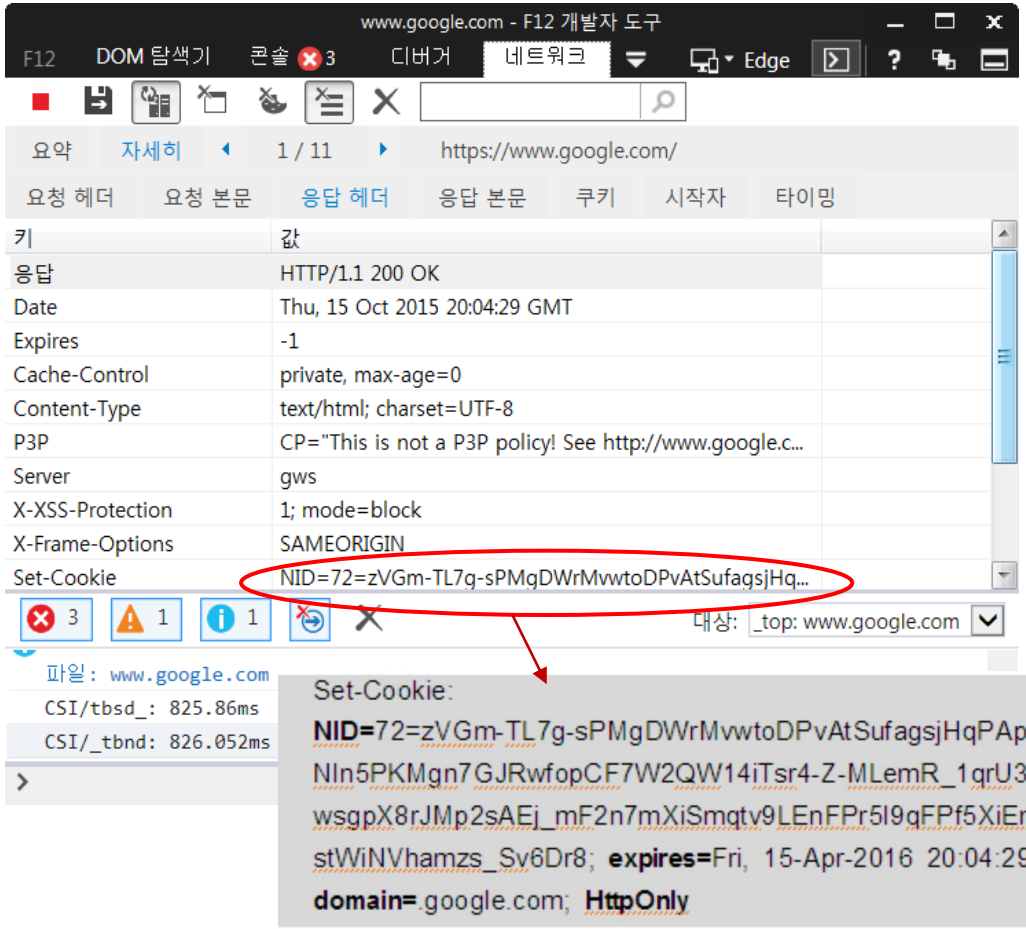
  - 브라우저가 google.com 사이트의 / 폴더에 있는 어떤 웹 자원이라도 요청할 때는 반드시 "age=23" 형태로 쿠키 전송
  - 유효 시간은 2016년 8월 1일까지, 안전한 통신을 사용할 때만 쿠키 사용
  - 구글 사이트와 HTTP 통신 외에 이 쿠키를 알려주어서는 안 됨

쿠키는 웹 페이지 사이의 정보 공유에 활용



실습2 : 구글 웹 사이트가 남긴 쿠키 보기

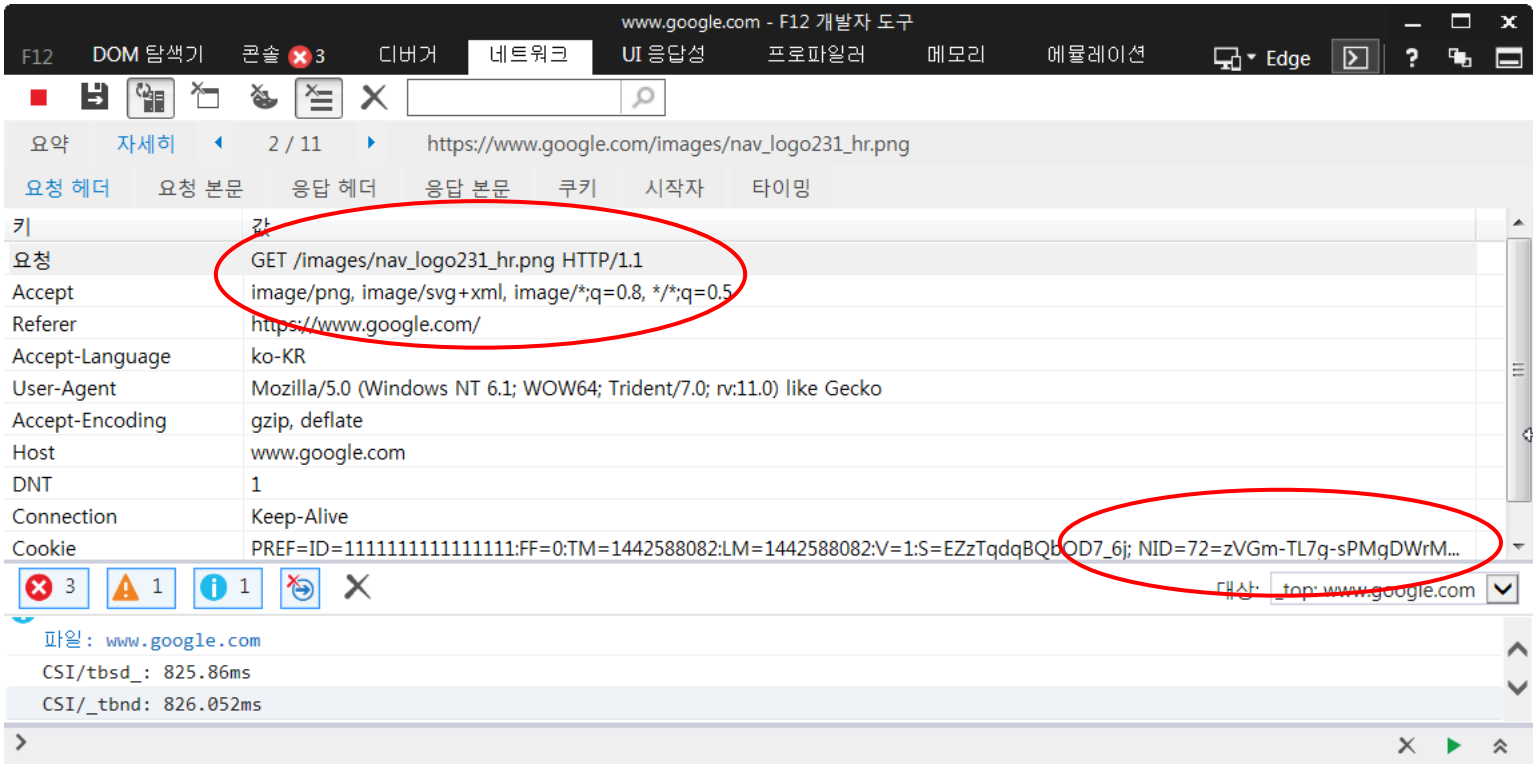
- 구글 웹 서버의 쿠키 뿌리기
  - 웹 서버는 HTTP 응답 헤더의 'Set-Cookie:' 뒤에 쿠키 데이터를 심어 보냄



NID 쿠키 저장 및 활용

- 브라우저는 NID 쿠키를 쿠키 파일에 저장
- 브라우저는 google.com 도메인의 웹 페이지를 요청할 때마다 NID 쿠키 함께 전송

브라우저가 nav_logo231_hr.png 이미지를 요청할 때,
www.google.com에 보낸 요청 헤더



자바스크립트로 쿠키 다루기

- 자바스크립트 코드를 이용하여 로컬 컴퓨터에 쿠키쓰기/읽기 가능
- 자바스크립트에서 쿠키 접근 : document.cookie
  - 윈도우에 출력된 웹 페이지를 전송한 웹 서버 모든 쿠키들이 문자열 형태로 연결
  - 쿠키 쓰기
    - document.cookie에 쿠키를 문자열 형태로 달아주면 됨

```
function SetCookie (name, value, expireDate) {
   var cookieStr = name + "=" + escape(value) +
      ((expireDate == null)?"":("; expires=" + expireDate.toGMTString()));
      document.cookie = cookieStr; // 쿠키를 연결하는 방식으로 저장
}
```

  - 쿠키 읽기

```
function GetCookie (name) {
   var str = name+"=";
   var pairs = document.cookie.split(";"); // 쿠키문자열을 ;을 경계로 분할
   for(var i=0; i<pairs.length; i++) {
      var pair = pairs[i].trim(); // 쿠키 앞뒤의 빈칸 제거
      var unit = pair.split("=");
      if(unit[0] == name)
         return unescape(unit[1]);
   }
   return null;
}
```

실습 3 : 쿠키 활용

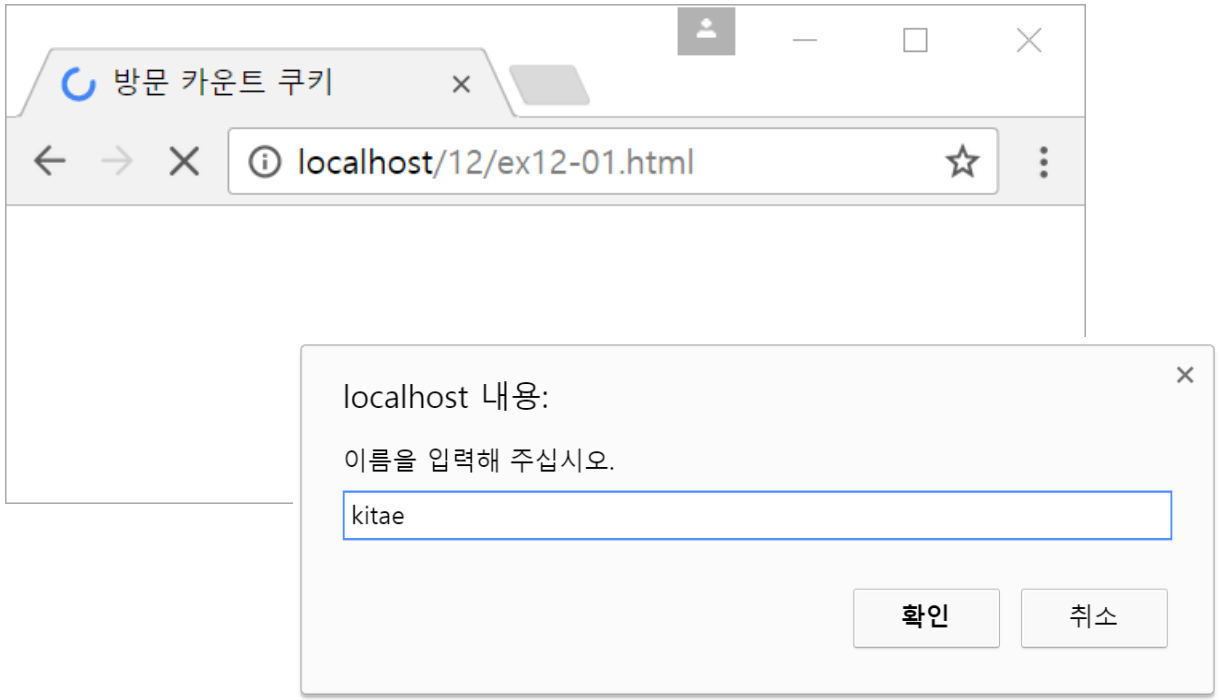
- 실습 전 준비 사항
  - 브라우저에 따라 다른 점
    - 익스플로러 경우, 웹 서버로부터 로드한 웹 페이지의 자바스크립트 코드에서만 쿠키 읽기/쓰기 가능
    - 크롬에서는 로컬 컴퓨터에서도 가능
  - 로컬 컴퓨터에 웹 서버 설치
    - 사용자 컴퓨터에 몽구스 웹 서버 다운 및 설치
    - 몽구스 사이트 에서 몽구스(mongoose-free-6.5.exe) 다운로드
      - https://www.cesanta.com/products/binary
      - http://www.webprogramming.co.kr
    - 설치 및 작동 방법은 부록 참고(매우 간단함)
- 자바스크립트로 방문자 이름과 방문 횟수 관리 실습
  - 실습 과정
    - ex12-01.html을 몽구스 웹 서버(예 :C:/webserver/12/) 폴더에 저장
    - ex12-01.html을 출력 http://localhost/12/ex12-01.html

예제12-1 쿠키 활용 : 자바스크립트로 방문자 이름과 방문 횟수 관리

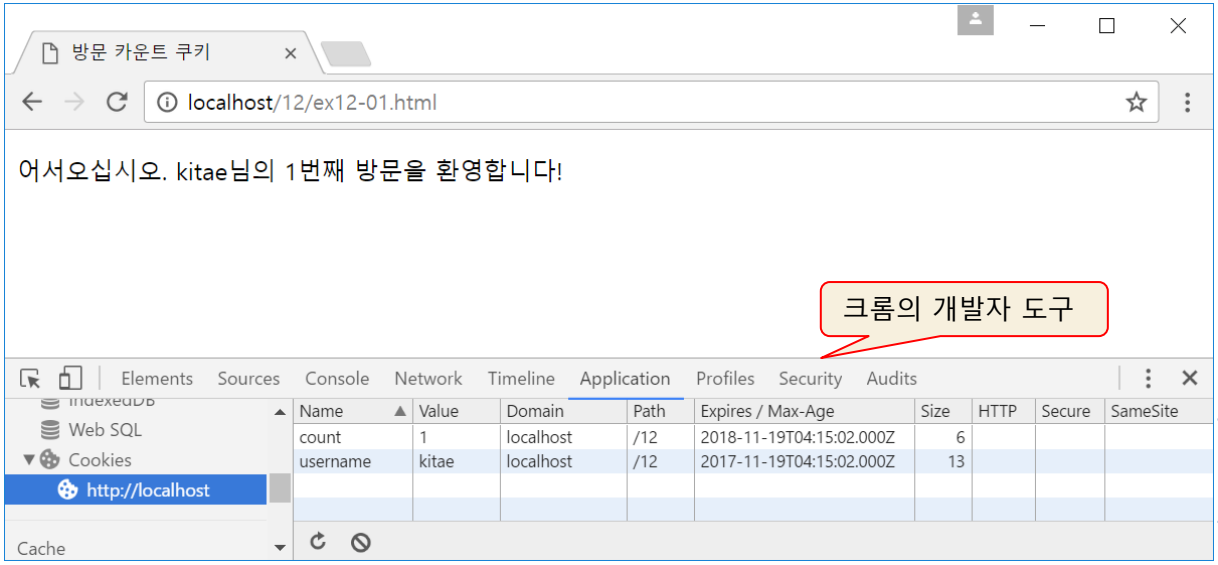

```
!DOCTYPE html>
<html>
<head><title>방문 카운트 쿠키</title>
<script>
function GetCookie (name) {
    var str = name+"=";
    var pairs = document.cookie.split(";"); // 쿠키문자열을 ;을 경계로 분할
    for(var i=0; i<pairs.length; i++) {
        var pair = pairs[i].trim(); // 쿠키 앞뒤의 빈칸 제거
        var unit = pair.split("=");
        if(unit[0] == name)
            return unescape(unit[1]);
    }
    return null;
}
function SetCookie (name, value, expireDate) {
    var cookieStr = name + "=" + escape(value) +
((expireDate == null)?"":("; expires=" + expireDate.toGMTString()));
document.cookie = cookieStr;
}
</script></head>
```

```
<body>
<script>
var username = GetCookie("username");
var count = GetCookie("count");
var expire = new Date (); // 현재 시간
if (username == null) {
    count = 0;
    username = prompt("이름을 입력해 주십시오.","");
    if (username == null) {
        alert("이름을 입력하시면 보다 나은 서비스를 제공받을 수 있습니다.");
        username = "아무개";
    } else {
        expire.setTime(expire.getTime() + (365 * 24 * 3600 * 1000)); // 1년후
        SetCookie("username",username,expire);
    }
}
count++;
expire.setTime(expire.getTime() + (365 * 24 * 3600 * 1000)); // 1년후
SetCookie("count",count,expire);
document.write('<p>어서오십시오. '+username+'님의 '+count+'번째
방문을 환영합니다!');
</script>
</body>
</html>
```

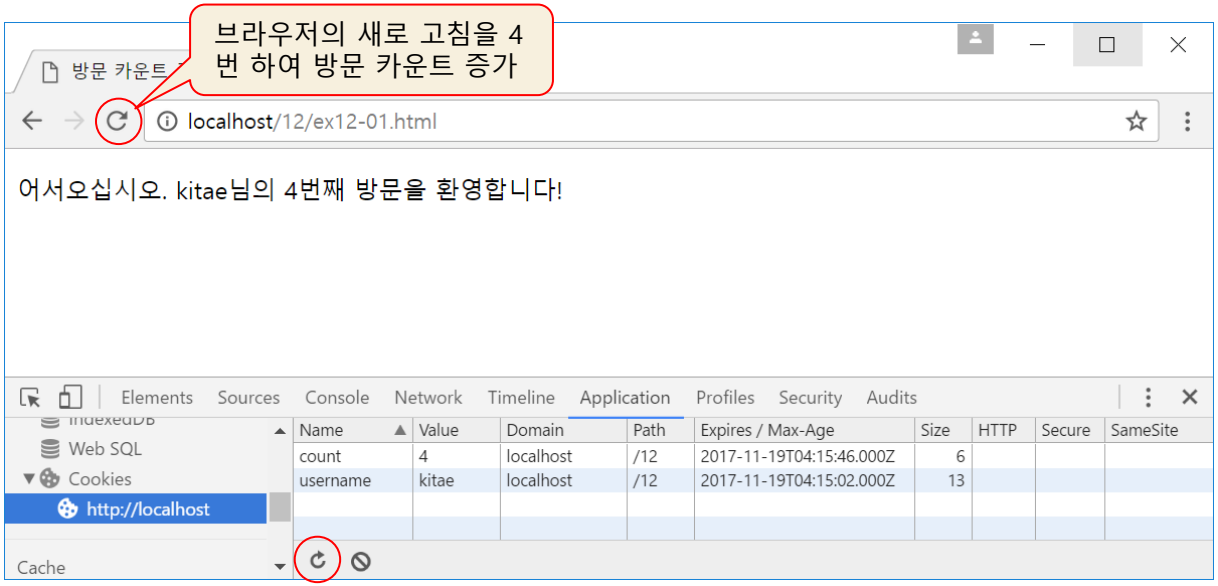
예제12-1 실행


방문 카운트 쿠키
localhost/12/ex12-01.html
localhost 내용:
이름을 입력해 주십시오.
kitae
확인
취소

예제 12-1 실행 결과



(a) 처음 방문한 경우(개발자도구로 쿠키보기)



(b) 새로 고침을 3번 하여 방문 회수가 증가한 경우(개발자도구로 쿠키보기)

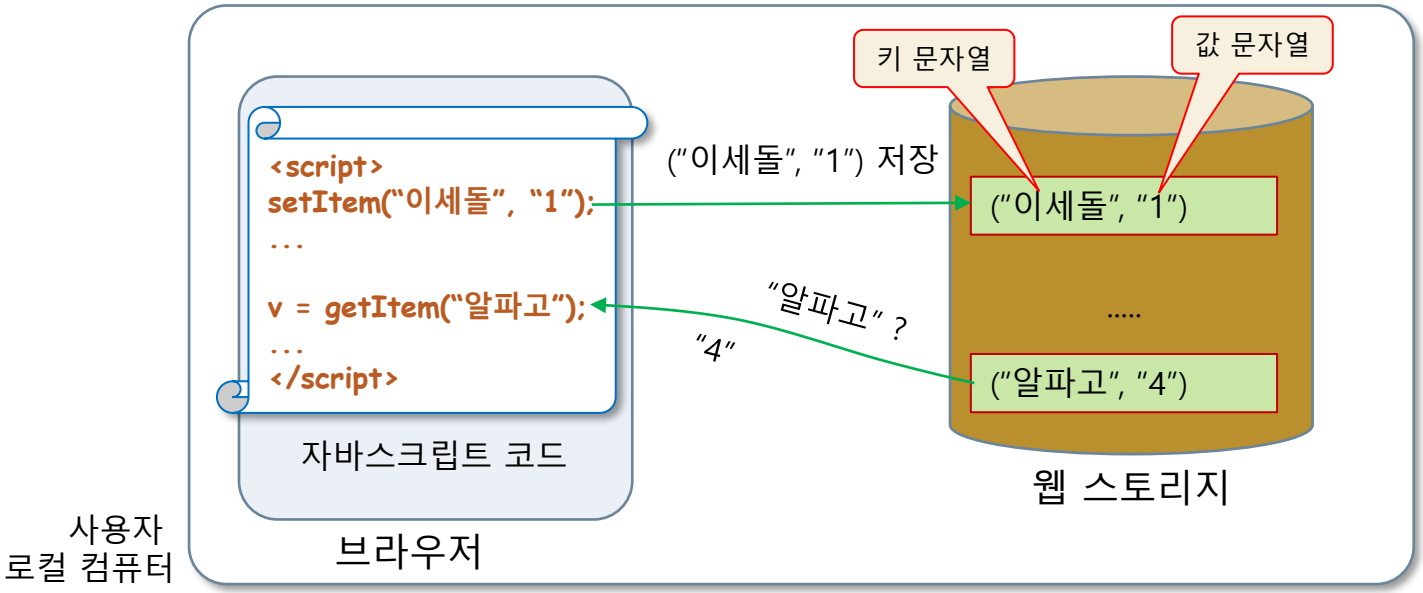
웹 스토리지(Web Storage)

- 웹 스토리지 필요성
  - 웹은 웹 애플리케이션의 형태로 진화
    - 웹 문서를 보여주기나 검색, 구매 등 정보 소통 수단을 넘어섬
    - 웹 애플리케이션 사례) 게임, 그림 그리기, 학습
  - 웹 애플리케이션은 실행 도중 로컬 컴퓨터에 데이터 저장 공간 필요
    - 예) 게임 웹 애플리케이션 : 사용자 이름, 점수, 최고 점수자의 이름과 점수 등
    - 예) 쇼핑몰 : 사용자가 구입하려고 담은 리스트
  - HTML5에서 웹 스토리지(web storage) 도입
    - 사용자 로컬 컴퓨터의 저장 공간
    - 웹 서버의 저장 부담과 네트워크 트래픽 감소
    - HTML5 웹 스토리지는 오직 자바스크립트로만 읽고 쓸 수 있음

- 쿠키의 한계
  - 쿠키의 크기는 4KB로 제한- 충분한 량의 정보 저장 한계
  - 쿠키는 불필요한 트래픽 발생
    - 브라우저가 웹 서버에 요청을 보낼 때마다 함께 전송하기 때문
  - 쿠키는 윈도우마다 독립적인 값을 저장 불가
    - 브라우저의 모든 윈도우들이 공유하므로

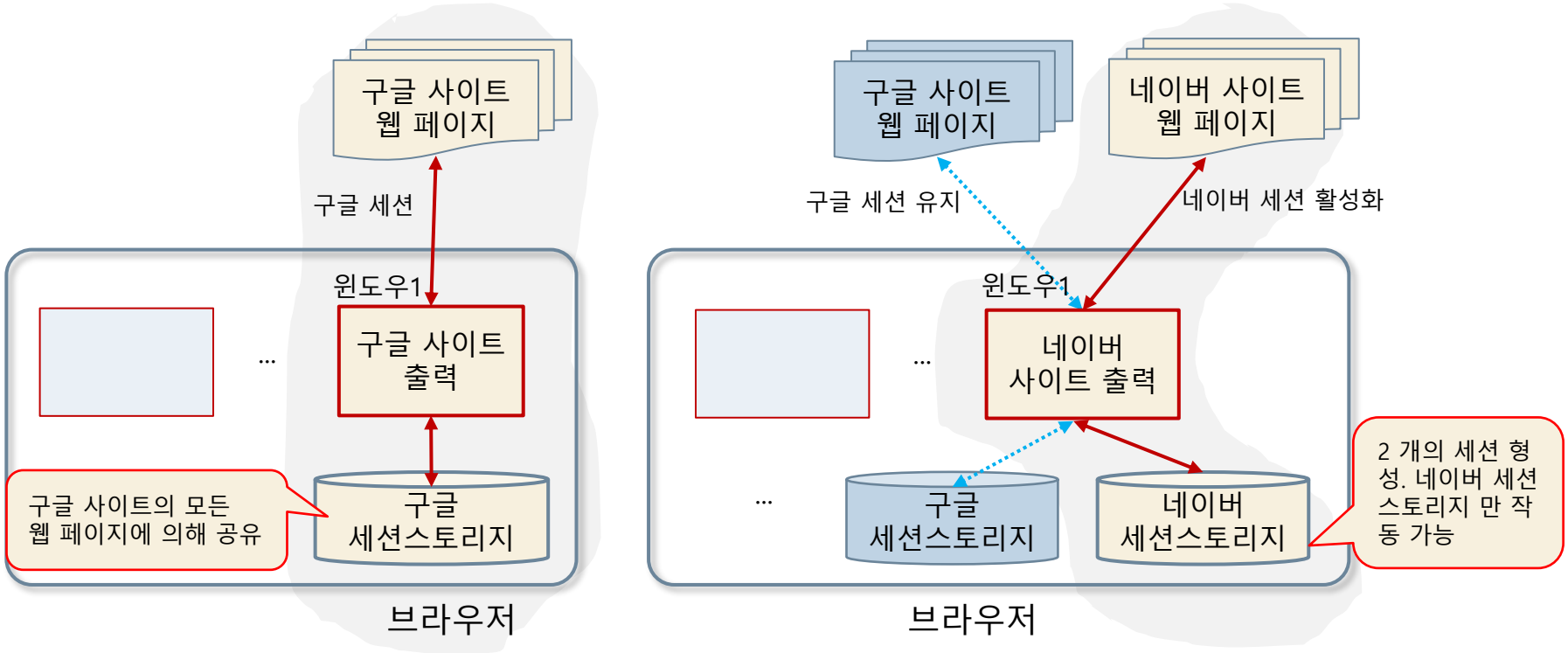
웹 스토리지 종류와 특징

- 웹 스토리지 : 세션스토리지(session storage)와 로컬스토리지(local storage)
- 웹 스토리지의 특징
  - 문자열만 저장
  - (키, 값)으로 구성된 아이템 단위로 저장
  - 동일한 '키'를 가진 아이템은 존재할 수 없음
  - '키'와 '값' 문자열은 대소문자 구분
  - 저장, 검색, 삭제 등 웹 스토리지의 조작은 모두 자바스크립트 코드로 작성



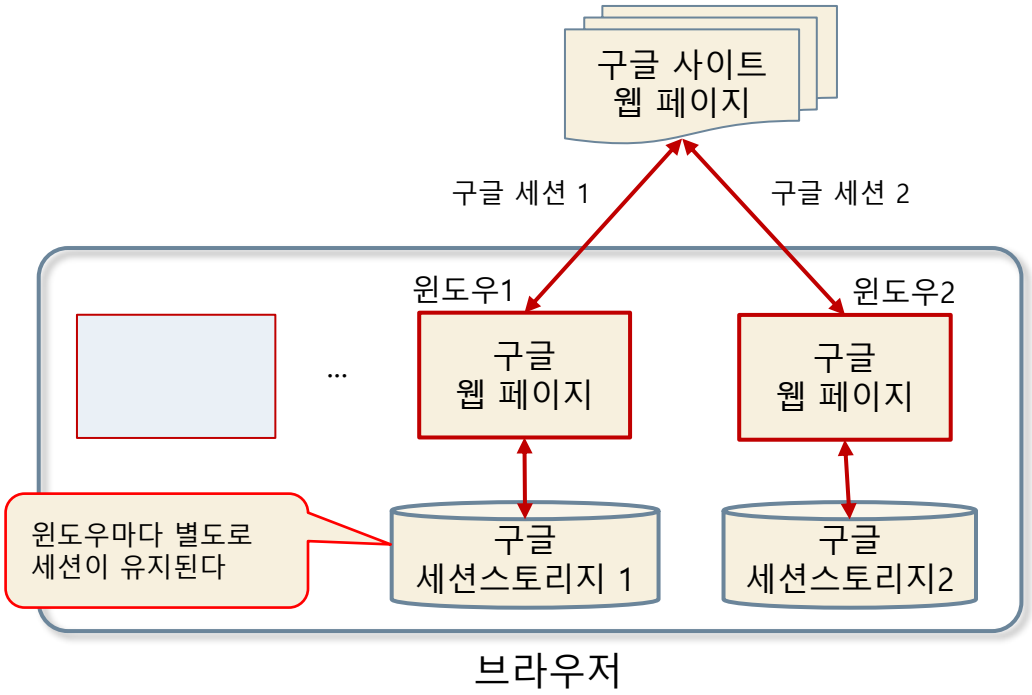
세션 스토리지

- 세션 스토리지의 생성과 소멸
  - 세션 : 연결된 웹 사이트와 윈도우
  - 윈도우에 웹 사이트가 로드될 때 세션 스토리지 생성
    - 한 윈도우에 여러 세션 스토리지 생성 가능
  - 윈도우가 닫힐 때 세션 스토리지 소멸



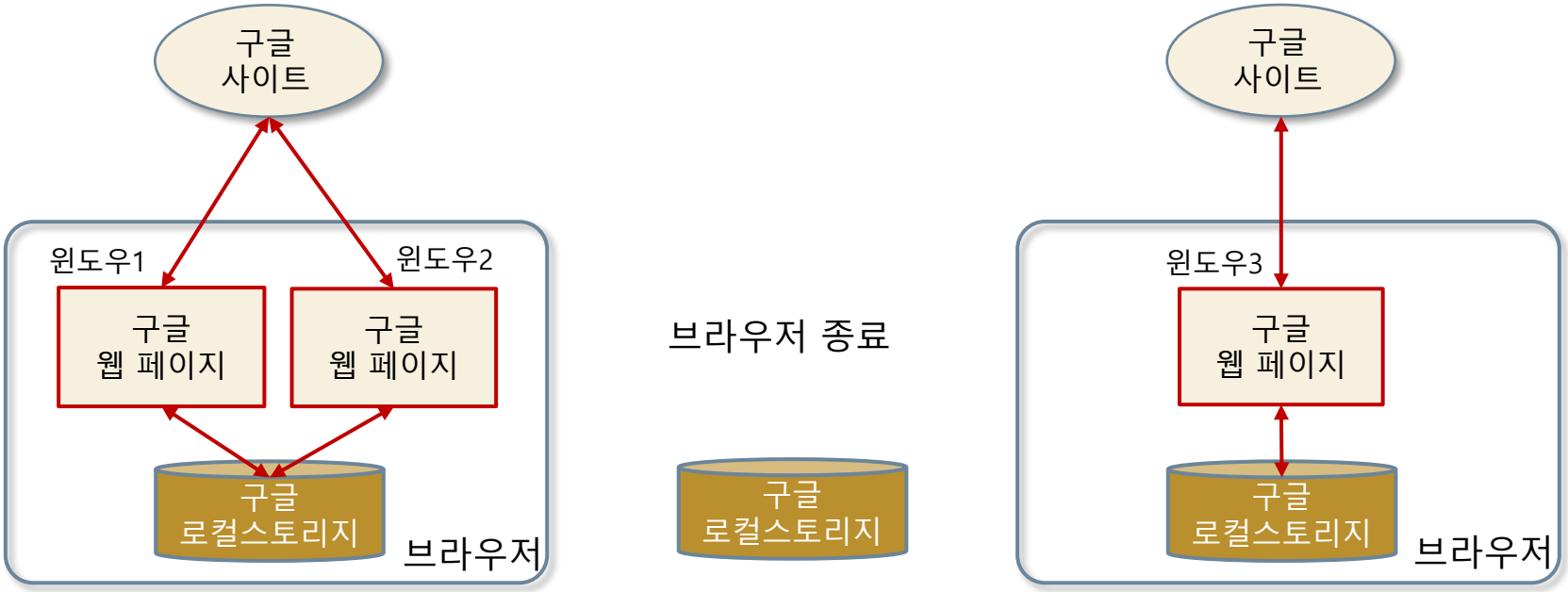
세션 스토리지의 공유 범위

- 윈도우마다 세션 스토리지 별도 생성
  - 윈도우 사이에서는 공유되지 않음
- 세션 스토리지 공유
  - 윈도우에 로드된 웹 사이트의 모든 웹 페이지들이 세션스토리지 공유
- 세션 스토리지의 용도
  - 한 윈도우에서 연결된 웹 사이트의 웹 페이지들끼리 데이터를 주고 받는 임시 저장 공간



로컬 스토리지

- 로컬 스토리지의 생성과 소멸, 공유
  - 윈도우에 상관없이 웹 서버(웹 사이트) 당 하나 씩 생성
  - 브라우저 종료하거나 컴퓨터가 커져도 존재
  - 웹 사이트의 모든 웹 페이지가 로컬 스토리지 공유
- 로컬 스토리지의 용도
  - 오프라인상태에서 웹 애플리케이션이 로컬 컴퓨터의 로컬 스토리지에 저장 가능



자바스크립트로 웹 스토리지 다루기

- Storage 인터페이스 : 자바스크립트로 웹 스토리지 읽기/쓰기
  - 브라우저가 제공

프로퍼티	설명	r/w
length	스토리지에 저장된 아이템의 개수	r
[]	['키']로 아이템의 '값'을 읽거나 저장하는 연산자	r/w

메소드	설명
key(index)	index 위치에 저장된 아이템의 '키' 문자열 반환
setItem(key, val)	(key, val) 아이템을 스토리지에 저장. key와 val 모두 문자열
getItem(key)	문자열 key의 아이템을 찾아 '값' 문자열 리턴. 아이템 없으면 null 리턴
removeItem(key)	문자열 key의 아이템 삭제
clear()	스토리지의 모든 아이템 삭제

  - 윈도우에 웹 페이지가 로드되면, 세션 스토리지와 로컬 스토리지 자동 생성
    - sessionStorage, localStorage, window.sessionStorage, window.localStorage
    - Storage 인터페이스의 프로퍼티와 메소드
    - 자바스크립트 코드로 웹 스토리지 액세스를 위한 객체

sessionStorage와 localStorage 다루기

- 웹 스토리지 지원 확인

```
if(!window.sessionStorage) {
    // 브라우저가 세션 스토리지 지원 않음
    alert("세션 스토리지를 지원하지 않습니다.");
}
```

- 아이템 저장 및 변경 : setItem()이나 [] 연산자 이용

```
sessionStorage.setItem("score", "80"); // 세션스토리지에 ("score", "80") 아이템 저장
sessionStorage["score"] = "80"; // 위와 동일한 코드
```

- 아이템 읽기 : '키'로 getItem()이나 [] 연산자 이용
  - '키' 아이템이 없는 경우, getItem()은 null 리턴

```
var myScore = sessionStorage.getItem("score"); // myScore = "80"
```

- 아이템 삭제 : removeItem() 이용

```
sessionStorage.removeItem("score"); // 세션 스토리지에서 "score" 키 아이템 삭제
```

- 모든 아이템 삭제 : clear()를 이용

```
sessionStorage.clear(); // 세션 스토리지의 모든 아이템 삭제
```

세션 스토리지의 모든 아이템 출력

- 세션 스토리지의 모든 아이템 출력

```
for(var i=0; i<sessionStorage.length; i++) {
    var key = sessionStorage.key(i);
    var val = sessionStorage.getItem(key);
    document.write(key + " " + val + "<br>");
}
```

실습 4 : 세션 스토리지 응용 실습

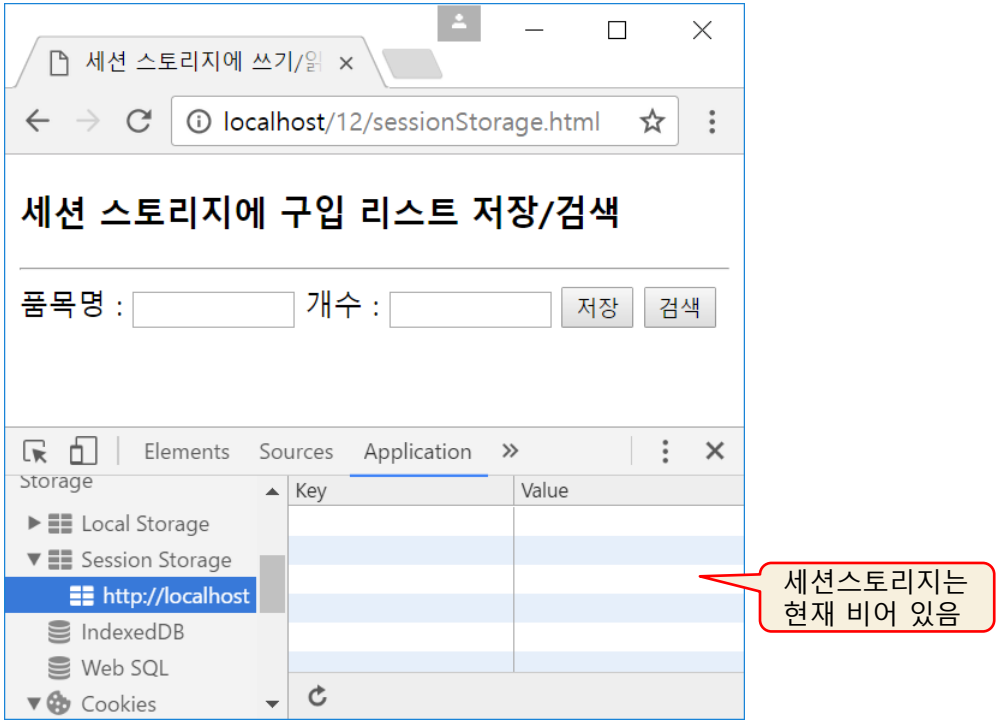
- Chrome 브라우저로 세션 스토리지에 아이템 저장/ 검색
- 1. 세션 스토리지를 조작하는 웹 페이지 작성

```
<!DOCTYPE html>
<html>
<head><title>세션 스토리지에 쓰기/읽기</title></head>
<body>
<h3>세션 스토리지에 구입 리스트 저장/검색</h3>
<hr>
품목명 : <input id="item" type="text">
개수 : <input id="count" type="text">
<button id="save" onclick="store()">저장</button>
<button id="retrieve" onclick="retieve()">검색</button>
<script>
var item = document.getElementById("item");
var count = document.getElementById("count");

function store() { // e는 이벤트 객체
    if(!window.sessionStorage) {
        alert("세션 스토리지를 지원하지 않습니다.");
        return;
    }
    sessionStorage.setItem(item.value, count.value);
}
```

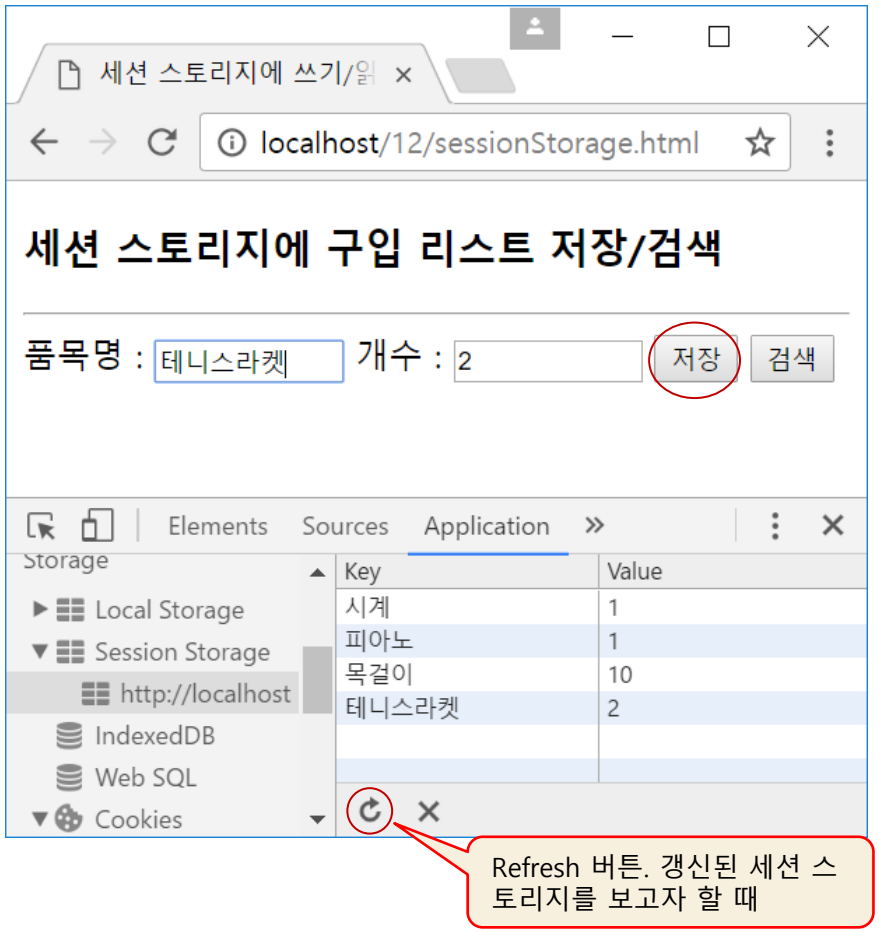
```
function retrieve() { // e는 이벤트 객체
    if(!window.sessionStorage) {
        alert("세션 스토리지를 지원하지 않습니다.");
        return;
    }
    var val = sessionStorage.getItem(item.value);
    if(val == null)
        alert(item.value + "는 구입 리스트에  없습니다.");
    else
        count.value = val;
}
</script>
</body>
</html>
```

sessionStorage.html

2. 개발자 도구로 세션 스토리지 보기

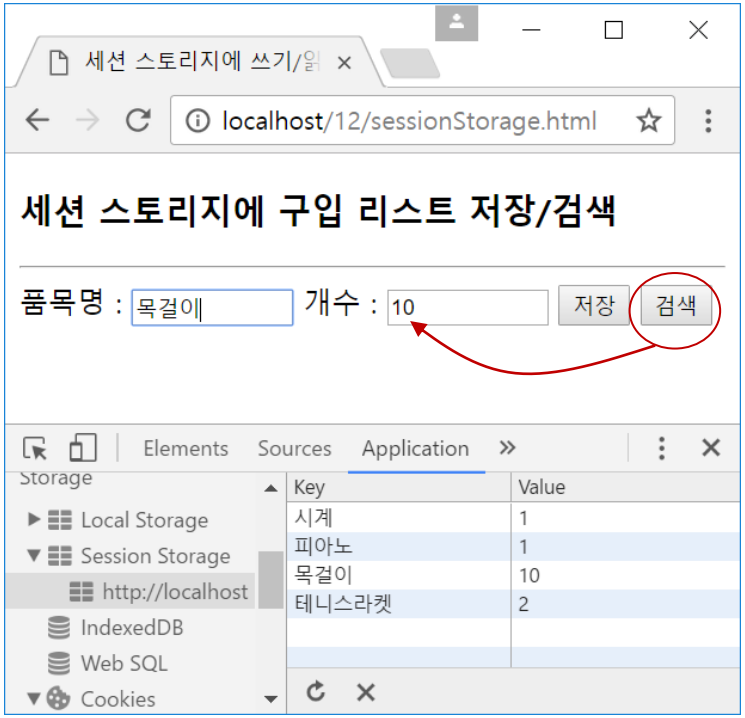


3. 세션 스토리지에 아이템 쓰기



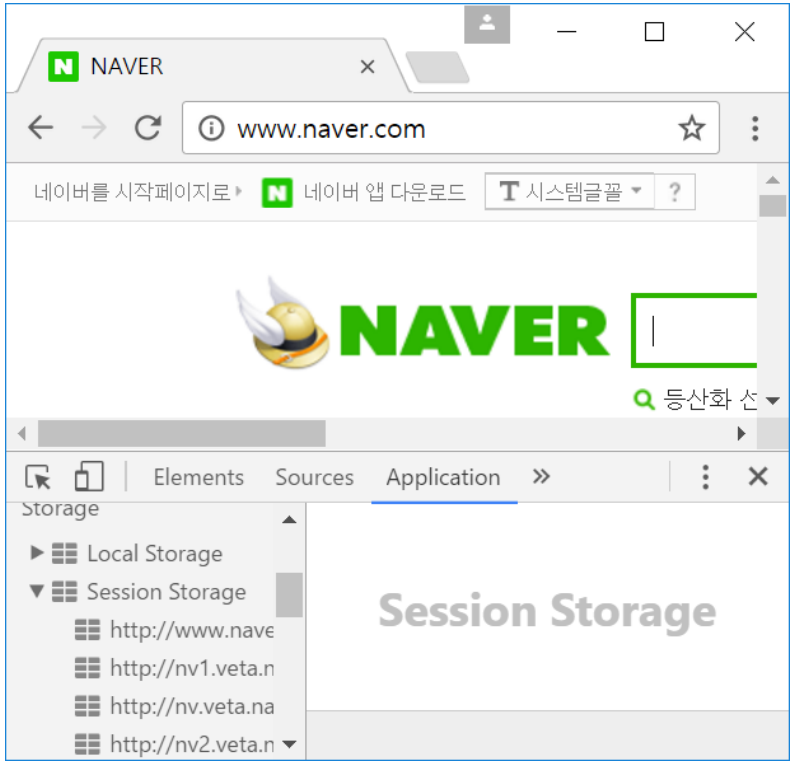

```
var item = document.getElementById("item");
var count = document.getElementById("count");

function store() { // e는 이벤트 객체
    if (!window.sessionStorage) {
        alert("세션 스토리지를 지원하지 않습니다.");
        return;
    }
    sessionStorage.setItem(item.value, count.value);
}
```

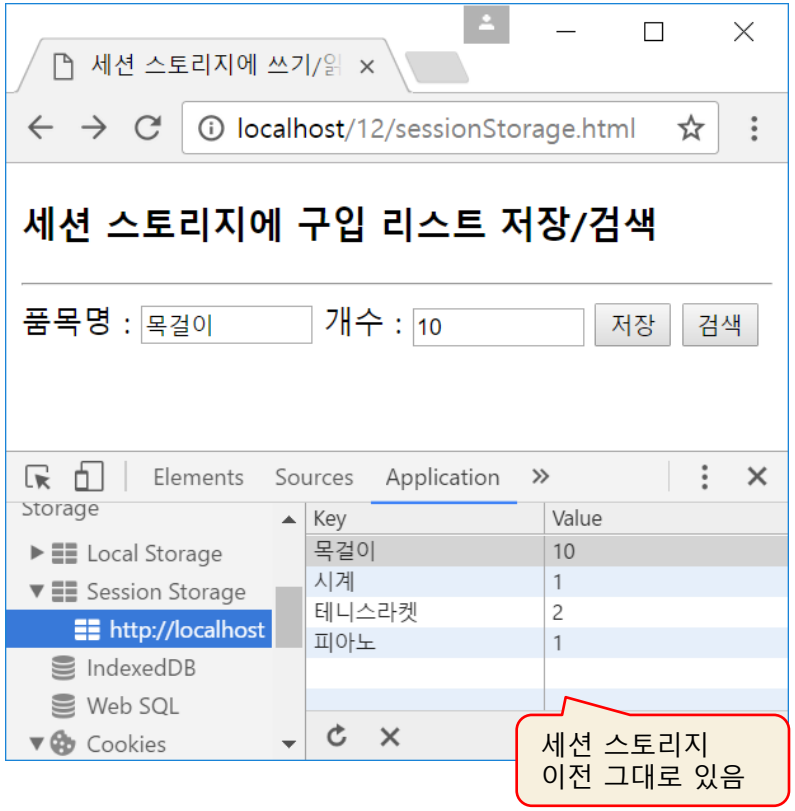
4. 세션 스토리지에서 아이템 검색




```
var item = document.getElementById("item");
var count = document.getElementById("count");

function retrieve() {
    if (!window.sessionStorage) {
        alert("세션 스토리지를 지원하지 않습니다.");
        return;
    }
    var val = sessionStorage.getItem(item.value);
    if(val == null)
        alert(item.value + "는 구입 리스트에 없습니다.");
    else
        count.value = val;
}
```

5. 다른 웹사이트 방문 후 돌아와 세션 스토리지 확인





웹 스토리지 이벤트

- storage 이벤트 : 웹 스토리지 변경 시 발생
  - 아이템 추가, 삭제, 전체 삭제, 아이템 값 변경 등의 경우
  - 웹 스토리지를 변경한 윈도우 외 다른 모든 윈도우에게 전달
  - window 객체만 storage 이벤트 받을 수 있음
- StorageEvent 객체 : 웹 스토리지 변경 정보를 담은 이벤트 객체

프로퍼티	설명	r/w
key	변화가 발생한 아이템의 키 문자열. clear()의 실행으로 이벤트가 발생한 경우는 null	r
newValue	변화가 발생한 아이템의 새 '값'. clear()의 실행이나 아이템이 삭제되어 이벤트가 발생한 경우는 null	r
oldValue	변화가 발생한 아이템의 이전 '값'. clear() 메소드가 호출되어 발생한 경우나 새로운 아이템이 추가되어 발생한 경우는 null	r
storageArea	이벤트가 발생한 웹 스토리지 객체	r
url	이벤트를 유발한 웹 페이지의 URL	r

- storage 이벤트 처리

```
window.addEventListener("storage", storageEventListener, false); // 이벤트 리스너 등록
function storageEventListener(e) { // e는 storage 이벤트 객체
        // 이벤트 처리 루틴 작성
}
```

실습 5: 로컬 스토리지에 storage 이벤트 실습

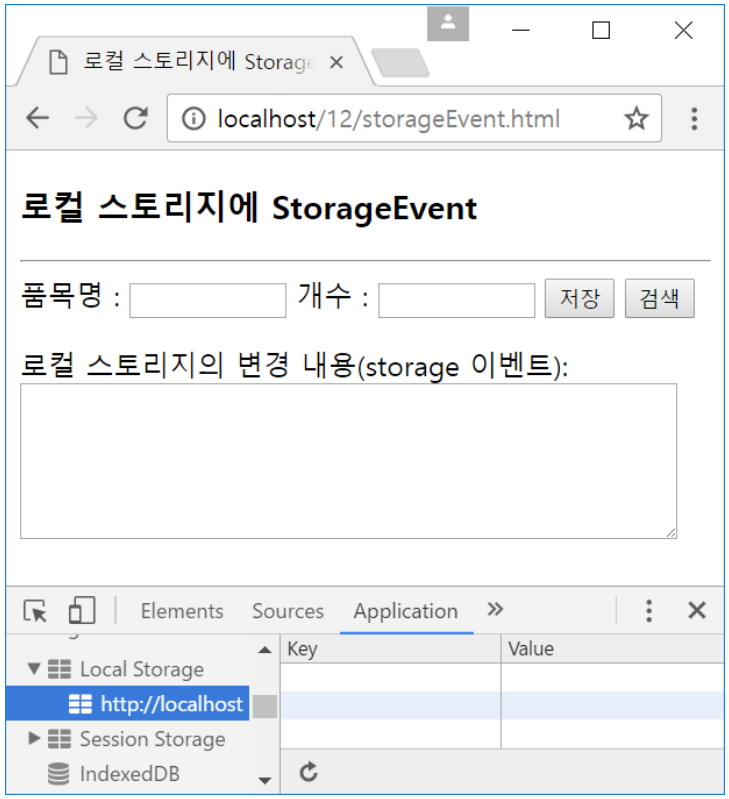
- storage 이벤트 실습 코드 : storageEvent.html

```
<!DOCTYPE html>
<html>
<head><title>로컬 스토리지에 StorageEvent</title>
</head>
<body>
<h3>로컬 스토리지에 StorageEvent</h3>
<hr>
품목명 : <input id="item" type="text" size="10">
개수 : <input id="count" type="text" size="10">
<button id="save" onclick="store()">저장</button>
<button id="retrieve" onclick="retrieve()">검색</button><p>
로컬 스토리지의 변경 내용(storage 이벤트):<br>
<textarea id="textarea" cols="60" rows="6"></textarea>

<script>
window.addEventListener("storage", storageEventListener, false);

function storageEventListener(e) { // e는 StorageEvent 객체
    var eventDetail = "key:₩t₩t₩t" + e.key + "₩n" +
            "oldValue:₩t₩t" + e.oldValue + "₩n" +
            "newValue:₩t₩t" + e.newValue + "₩n" +
            "storageArea:₩t" + e.storageArea + "₩n" +
            "url:₩t₩t₩t" + e.url;
    document.getElementById("textarea").innerHTML = eventDetail; //<textarea>에 출력
}
</script>
```

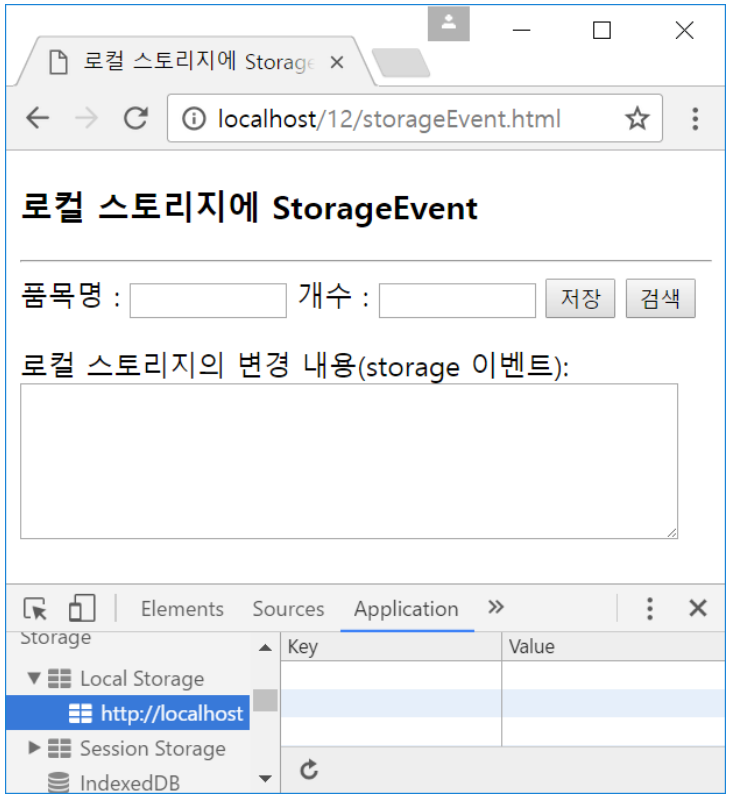
```
<script>
var item = document.getElementById("item");
var count = document.getElementById("count");

function store() {
    if (!window.localStorage) {
        alert("로컬스토리지를 지원하지 않습니다.");
        return;
    }
    localStorage.setItem(item.value, count.value);
}
function retrieve() {
    if (!window.localStorage) {
        alert("로컬스토리지를 지원하지 않습니다.");
        return;
    }
    var val = localStorage.getItem(item.value);
    if(val == null)
        alert(item.value + "는 구입 리스트에  없습니다.");
    else
        count.value = val;
}
</script>
</body>
</html>
```

1. Chrome 브라우저로 두 윈도우에 각각 storageEvent.html 열기

http://localhost/12/storageEvent.html

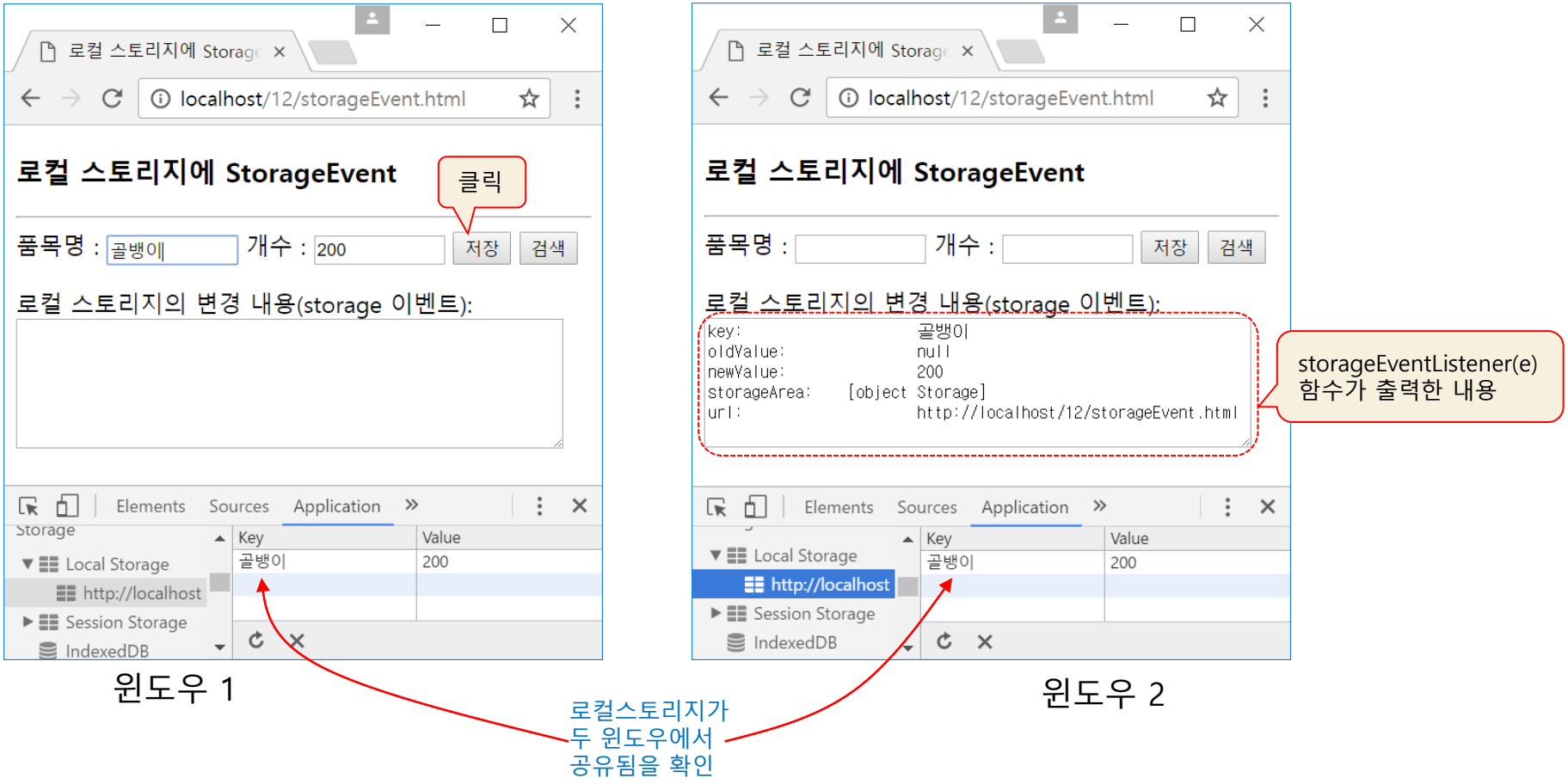


윈도우 1



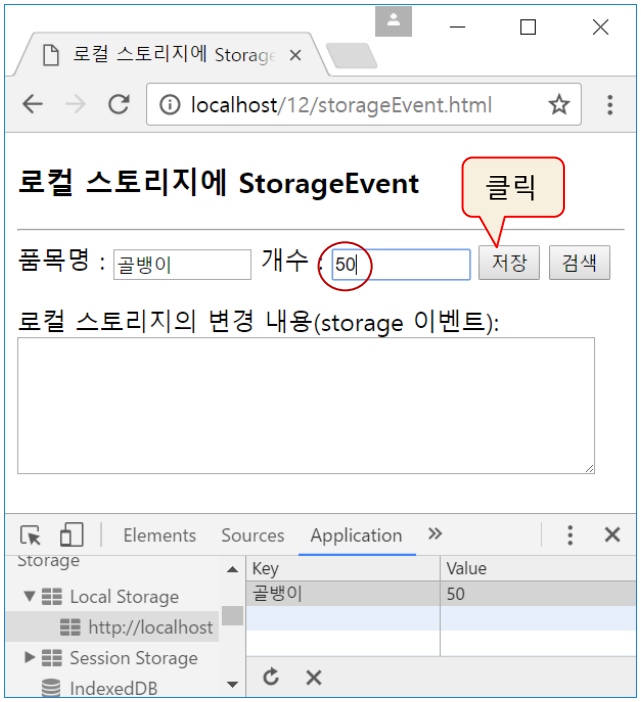
윈도우 2

2. 윈도우 1에서 (골뱅이, 200) 아이템을 로컬 스토리지에 저장

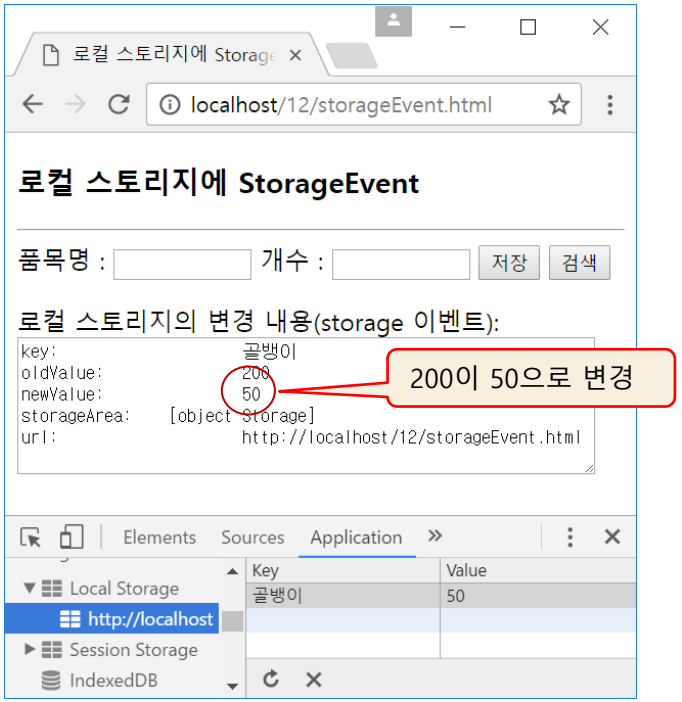


윈도우 2에 storage 이벤트 전달. 두 윈도우의 로컬스토리지 공유

3. 윈도우1에서 (골뱅이, 200) 아이템의 '값'을 200에서 50으로 수정



윈도우 1



윈도우 2

윈도우1에서 아이템을 변경하면 윈도우2에 storage 이벤트 발생