Web Storage

Offline Web Application 과 비슷하게 갑자기 전원이 꺼진다거나 다시 PC 를 켰을 때 이전에 작업한 데이터를 보존할 수도 있고, 웹 메일을 Web Storage 에 저장해 두었다가 읽는다거나, 서버의 많은 정보를 Client 에 저장해 둘 수 있습니다.

Web Stroage 는 Client 의 disk 에 소량의 데이터를 저장하기 위한 Storage 로 이전까지는 Cookie 를 사용했습니다만 Cookie 와는 다음과 같은 몇 가지 다른 점이 있습니다.

크기에 제한이 없음.	Cookie 는 4KB 로 제한이 있지만 Web Storage 는 제한이 없습니다.
서버로 보내지지 않음.	Cookie 는 HTTP Request 에 의해서 자동으로 서버에 전송이 되지만 Web Storage 는 서버로 전송되지 않습니다.
유효기간의 제한이 없음.	Cookie 처럼 특정기간이 지나면 자동으로 삭제되지 않습니다.
JavaScript 객체를 저장할 수 있음.	JavaScript 객체를 저장할 수 있습니다.

Web storage 는 Web Browser 마다 별도로 가지고 있는 저장 공간에 Data 를 Key-Value 형식으로 저장합니다.

Web Storage 는 용도에 따라서 Local Storage 와 Session Storage 로 나뉘는데, 두 Storage 의 차이는 저장기간이나 유효범위만 다를 뿐 거의 같은 API 를 사용합니다.

그럼 Local Storage 부터 살펴보도록 하겠습니다.

1. Local Storage

Local Storage 는 저장기간에 제약이 없고 Web Site 의 Domain 에 따라서 생성되는 Storage 입니다.
저장기간도 사용자가 삭제하기 전까지 영구적으로 저장할 수 있습니다. 그렇기 때문에 기존의 Cookie 를 이용한 저장작업을 Local Storage 가 대신하기에 충분합니다. 거기다

Server 측과의 통신작업도 없기 때문에. Server 측에 부하도 주지 않게 됩니다.

그럼 그 사용법을 봅니다.
먼저 Web Browser 가 Local Storage 를 지원하는지 Check 해 봐야 할 것입니다.

```
<script type="text/javascript">
  if(!window.localStorage) {
    document.write('이 Browser 는 Local Storage 를 지원하지 않습니다.');
  } else {
    .....
  };
</script>
```

이젠 저장하는 방법을 보겠습니다.

```
<script type="text/javascript">
  if(!window.localStorage) {
    document.write('이 Browser 는 Local Storage 를 지원하지 않습니다.');
  } else {
    localStorage.setItem("NickName", "woojja");
  };
</script>
```

저장을 했으니, 다시 가져와 봅니다.

```
<script type="text/javascript">
  if(!window.localStorage) {
    document.write('이 Browser 는 Local Storage 를 지원하지 않습니다.');
  } else {
    localStorage.setItem("NickName", "woojja");
    alert(localStorage.getItem("NickName"));
  };
</script>
```

삭제하고자 한다면 다음과 같이 합니다.

```
<script type="text/javascript">
  if(!window.localStorage) {
     document.write('이 Browser 는 Local Storage 를 지원하지 않습니다.');
  } else {
     localStorage.setItem("NickName", "woojja");
     alert(localStorage.getItem("NickName"));
     localStorage.removeItem("NickName");
  };
</script>
```

하나씩 지우지 않고, 한번에 모두 삭제하려면

```
<script type="text/javascript">
  if(!window.localStorage) {
     document.write('이 Browser 는 Local Storage 를 지원하지 않습니다.');
  } else {
     localStorage.setItem("NickName", "woojja");
     alert(localStorage.getItem("NickName"));
     localStorage.removeItem("NickName");
     localStorage.clear();
  };
</script>
```

이번엔 조금 다른 방법을 살펴보도록 하겠습니다.
저장할 때는

```
localStorage.setItem("NickName", "woojja");
localStorage.NickName = "woojja";
localStorage["NickName"] = "woojja";
```

가져올 때는

```
var NickName = localStorage.getItem("NickName");
var NickName = localStorage.NickName;
var NickName = localStorage["NickName"];
```

지울 때는

```
localStorage.removeItem("NickName");
delete localStorage.NickName;
delete localStorage["NickName"];
```

와 같은 방법으로 사용을 합니다.

전체 Storage Data 에 대해서

```
for(var i=0; i<localStorage.length; i++) {
var key = localStorage.key(i);
var value = localStorage[key];
...
}
```

또는

```
for(var key in localStorage) {
var value = localStorage[key];
...
}
```

이렇게 접근해서 사용할 수 있습니다.

Session Storage 도 사용 방법은 다르지 않습니다.
내용에 약간의 차이가 있을 뿐인데, 이번엔 Session Storage 를 살펴보도록 하겠습니다.

2. Session Storage

Session Storage 는 Local Storage 와 같이 도메인마다 생성이 되지만 Browser Window 의 유효범위와 생존기간을 갖습니다.
그래서 Local Storage 와는 다르게 Browser Window 가 닫히면 Session Storage 도 같이 삭제됩니다.
그럼 Browser Window 를 복제하면 어떻게 될까요?
예상은 하고 계시겠지만, 복제된 순간까지는 Session Storage 의 내용은 같겠지만 그 이후의 내용은 서로에게 아무런 영향을 주지도 않고 별개의 Storage 로 작동하게 됩니다.

이런 특징을 갖는 Session Storage 는 사용자의 동작을 추적하여 기록할 때 사용할 수 있다고 합니다.

Session Storage 의 사용은 Local Storage 와 같다고 하였는데. 단지 localStorage 를 **sessionStorage** 객체로 바꿔 사용하시기만 하면 됩니다.

3. Storage Event Handling

Storage 는 Storage 의 변경에 대해서 window 객체가 Storage Event 를 발생시킵니다. 다음 표는 그 Event 가 가지고 있는 속성들 입니다.

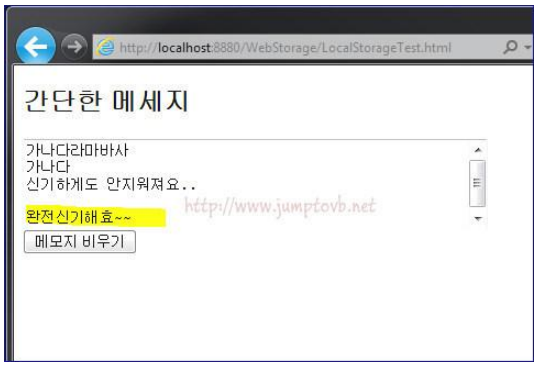
속성	설명
key	변경된 Key, clear() Method 가 호출되면 null 을 반환한다.
oldValue	변경되기 전의 값(복사본). 새로운 키로 값을 등록하였다면 null.
newValue	변경된 후의 값(복사본). 값이 삭제되었을 때는 null.
url	Event 가 발생한 문서의 URL.
storageArea	변경된 Storage 참조.

4. 예제

아래 예제는 LocalStorage 에 대한 내용입니다.

```
<!DOCTYPE html>
< html lang="ko">
    <head>
            <meta charset="utf-8" />
     <title>간단 메모지</title>
          <script type="text/javascript">
              function saveText() {
                  info = document.getElementById("info");
                  info.style.display = "block";
                  localStorage.setItem("memo", msg.value);
```

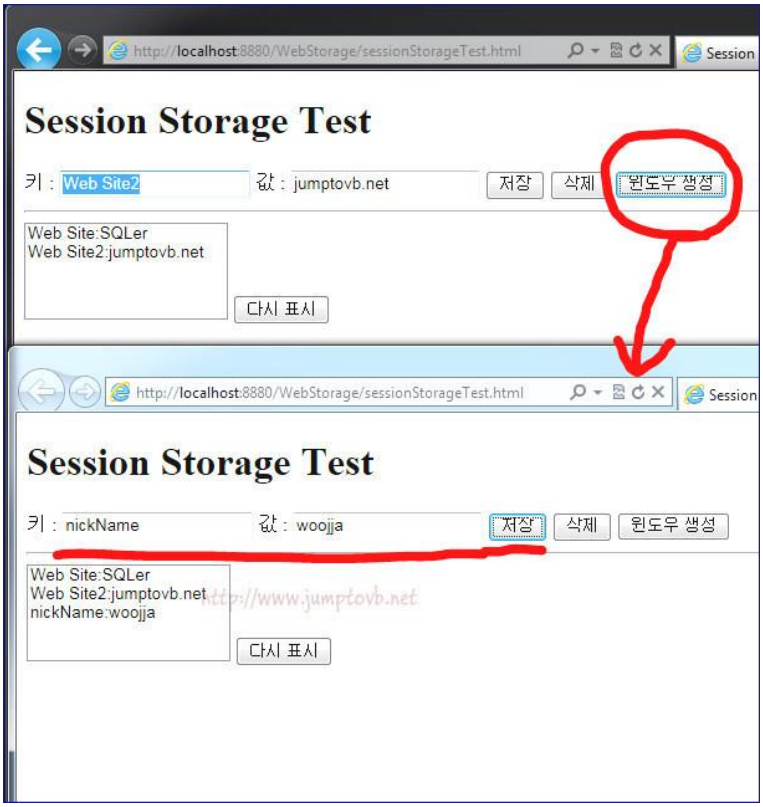
```
            };
            function pageload() {
                msg = document.getElementById("txtBox");
                msg.value = localStorage.getItem("memo");
            };
            function clr() {
                msg.value = "";
                localStorage.clear();
                info.style.display = "none";
            };
        </script>
    </head>
    <body onload="pageload()">
        <h2>간단한 메세지</h2>
          <textarea id="txtBox" onKeyDown="saveText();" onKeyUp="saveText();"
cols="50" rows="5"></textarea><br/>
        <input type="button" value="메모지 비우기" onClick="clr();" onKeyUp="clr();"
/>
        <p id="info" style="display:none;">메모내용이 자동 저장되었습니다.</P>
    </body>
< /html>
```



다음 예제는 session Storage 에 대한 내용입니다.
값을 두 개 넣고 난 후 윈도우 창을 복제하였습니다.
그리고 두 번째 생긴 Window 에서 값을 넣어 본 것입니다. 두 개의 storage 가 별개라는 것을 확인해 봅니다.

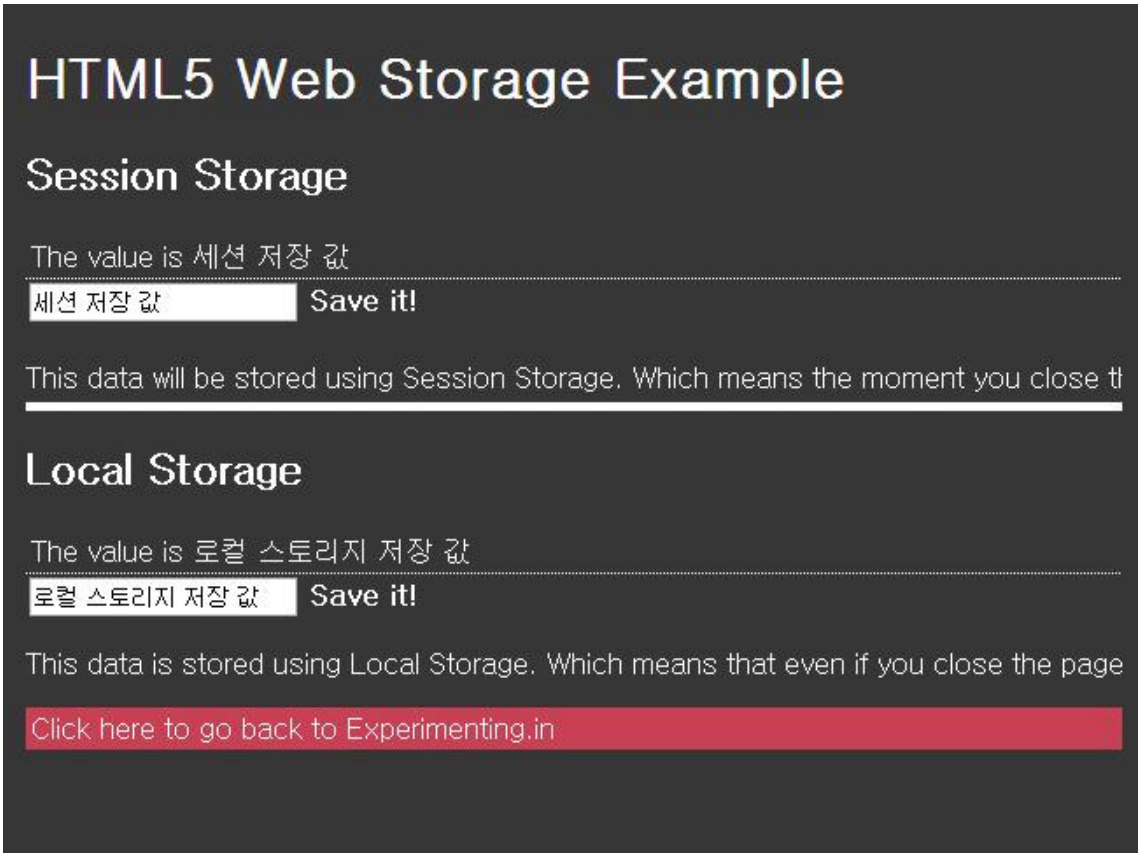
```
<!DOCTYPE html>
< html lang="ko">
    <head>
            <meta charset="utf-8" />
      <title>Session Storage Viewer</title>
     </head>
     <body onload="ShowAll()">
            <h1>Session Storage Test</h1>
            키 : <input id="k" type="text"> 값 : <input id="v" type="text">
            <button onclick="Save()">저장</button>
            <button onclick="Remove()">삭제</button>
            <button onclick="window.open(location.href);">윈도우 생성</button>
            <hr>
            <select id="entries" size=5 onchange="onEntrySelected()">
            </select>
            <button onclick="ShowAll()">다시 표시</button>

            <script type="text/javascript">

                var key = document.getElementById("k");
                var value = document.getElementById("v");
                var entries = document.getElementById("entries");

                function ShowAll() {
                    entries.innerHTML = "";
                    for (var i=0; i < sessionStorage.length; i++ )
                    {
                        var k = sessionStorage.key(i);
                         entries.options[entries.options.length] = new Option(k + ":" +
sessionStorage[k], k);
```

```
                }
            };

            function Save() {
                sessionStorage[key.value] = value.value;
                ShowAll();
            };

            function Remove() {
                delete sessionStorage[key.value];
                ShowAll();
            };

            function onEntrySelected() {
                var selectedOption = entries.options[entries.selectedIndex];
                key.value = selectedOption.value;
                value.value = sessionStorage[selectedOption.value];
            };
        </script>

    </body>
< /html>
```



"HTML5 API 로 추가된 Local Storage 와 Session Storage 의 사용법을 보여주면서
해당 창이 닫히면 기억이 사라지는 Session Storage 와
창이 닫혀도 기억이 남아있는 Local Storage 의 차이점을 보여주는 예제 입니다."

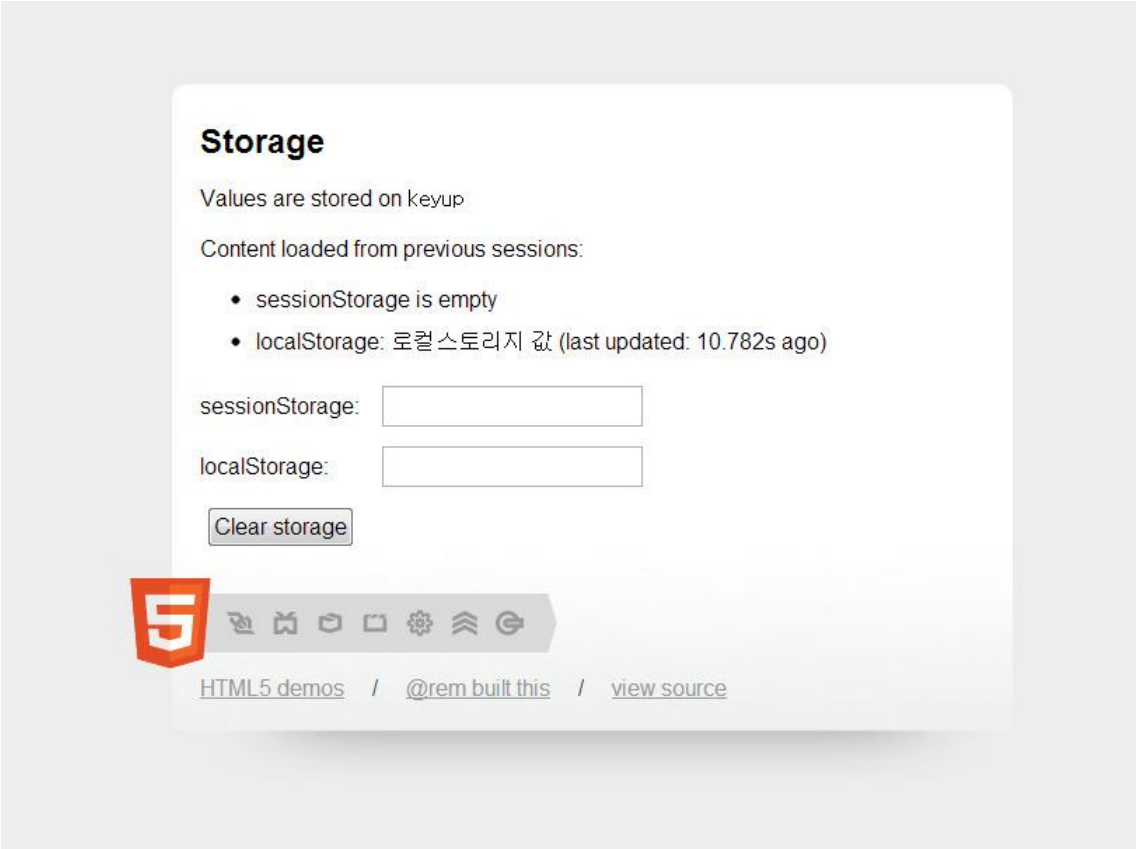
소스다운로드 : demo_webstorage.zip



"HTML5 API 로 추가된 Local Storage 와 Session Storage 의 사용법을 보여주면서 해당 창이 닫히면 기억이 사라지는 Session Storage 와 창이 닫혀도 기억이 남아있는 Local Storage 의 차이점을 보여주는 페이지 입니다."

소스다운로드 :

storage.zip



브라우저 수용버전 예) Chrome Ver. 6.0 이상의 버전에서 수용 가능

Chrome	Internet Explorer	Firefox	Safari	Opera
Ver. 6.0	Ver. 8.0	Ver. 3.6	Ver. 5.0	Ver. 10.60