10th week–
Javascript API

Web storage

- Definition
  - Data are needed to be stored/used in several web pages within the same domain
  - Before HTML5, cookie or session was used
  - Web storage is supported for storing data in the client in HTML5
- Cookie vs. Web storage
    - Cookie is transferred being embedded with each request and response
    - The size of cookie is restricted to 4K
    - Larger storage is supported
    - Transferred to server when needed

Web storage

- Interface(www.w3.org/TR/webstorage/)

```
interface Storage {
    readonly attribute unsigned long length;
    DOMString? key(unsigned long index);
    getter DOMString getItem(DOMString key);
    setter creator void setItem(DOMString key, DOMString value);
    deleter void removeItem(DOMString key);
    void clear();
};
```

- Objects supported
  - localStorage: data is stored after a window is closed
  - sessionStorage:
    - Data is stored together with window objects
    - Don't know the value of other window/tabs
    - Data is deleted when window/tab is closed

Web storage

- Way to use
  - to set value
    - localStorage.setItem('age', 40);
  - to get the stored value
    - var age = localStorage.getItem('age');
    - var iage= parseInt(age) + 3;
  - to remove the key/value pair from storage
    - localStorage.removeItem('age');
  - to clear all key/value pairs
    - localStorage.clear();
  - Key and length

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

Web storage

- The other usages
  - localStorage.age = 40 // set age value
  - var age = localStorage.age; // get age value
  - delete localStorage.age; // delete age value in storage

  - localStorage["sticky_0"] = "모바일응용 강의";
  - var sticky = localStorage["sticky_0"];

```
<html><head><title>Shell Game</title><meta charset="utf-8">
<script>
window.onload =shellGame;
function shellGame() {
    localStorage.setItem("shell1", "pea");
    localStorage.setItem("shell2", "empty");
    localStorage.setItem("shell3", "empty");
    localStorage["shell1"] = "empty";
    localStorage["shell2"] = "pea";
    localStorage["shell3"] = "empty";
    var value = localStorage.getItem("shell2");
    localStorage.setItem("shell1", value);
    value = localStorage.getItem("shell3");
    localStorage["shell2"] = value;
    var key = "shell2";
    localStorage[key] = "pea";
    key = "shell1";
    localStorage[key] = "empty";
    key = "shell3";
    localStorage[key] = "empty";

    for (var i = 0; i < localStorage.length; i++) {
        var key = localStorage.key(i);
        var value = localStorage.getItem(key);
        alert(key + ": " + value);
    }}
</script></head><body></body></html>
```

Practice 4

- Do you remember the exercise 2 program in 7th? Is there any problem to use?
  - After going out the program, what is happen to your songs list?
  - When you connect this program again, are there new songs added? -> no
  - One solution is localStorage
  - Whenever you add a song, it is stored to local storage. And when you connect the program, every stored songs are added automatically
- Compare the previous program with the program in the next slide

```
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>음악 추가 시키기</title>
   <script>
       window.onload= start;  //이전 데이터 가져오기
       function start(){
           for(var i = 0; i < localStorage.length; i++){//로컬스토리지 길이만큼 반복
                       var key = localStorage.key(i);
                       var li = document.createElement("li");
                       li.innerHTML = localStorage[key];
                       var ul = document.getElementById("playlist");
                       ul.appendChild(li);
           }
       }
       function addSong(){
           var input= document.getElementById("songName");
           var add= document.createElement("li");
           add.innerHTML=input.value;
           var ul= document.getElementById("playlist");
           ul.appendChild(add);
           var str = "list"+localStorage.length;
           localStorage.setItem(str,input.value);
       }
   </script>
</head>
```

Practice5

- The addsong program looks like good but it has a big problem?
- Do you know the problem?

- One solution to use array
- Try to solve!!!

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>음악 추가 시키기</title>
<script>
	window.onload= start;	//이전 데이터 가져오기
	function start(){
		var value = localStorage.getItem('localSonglist');
		if(value!=null){
			var storedStorage=JSON.parse(value);
			for(var i=0; i<storedStorage.length; i++){
				var li = document.createElement("li");
				li.innerHTML = storedStorage[i];
				var ol = document.getElementById("playlist");
				ol.appendChild(li);
			}
		}
	}
```

```
var Songlist = new Array();
function AddSong() {
        var input= document.getElementById("songName");
        if(input.value!=""){
                var li = document.createElement("li");
                li.innerHTML = input.value;
                var ol = document.getElementById("playlist");
                ol.appendChild(li);
                Songlist.push(input.value);
                localStorage.setItem("localSonglist", JSON.stringify(Songlist));
        }
}
</script> </head>
```

JSON

- 정의
  - JSON (JavaScript Object Notation)은 텍스트-기반의 데이터 형식
  - JSON은 자바 스크립트 언어에서 비롯됨
  - JSON 형식은 Douglas Crockford에 의하여 처음으로 지정되었으며, RFC 4627에 기술
  - 파일 이름 확장자는 .json
- 장점
  - 자바스크립트 객체와 값으로 빠르게 파싱 가능
  - 네트워크간 자료 교환이나 저장에 사용
  - XML이 뒤로 밀림

JSON 객체

- 주요 메쏘드
  - JSON.stringfy(Object_name)
  - JSON.parse(jsonString)
- 예제

```
var movie= {title: “덕혜옹주”,
            genre: “역사”,
            rating: 5,
            showtimes: [“15:00”, “19:00”, “21:00”]
};

var json_movie_s = JSON.stringfy(move);
alert(json_movie_s);

var json_movie_o = JSON.parse(json_movie_s);
alert(json_movie_o.title);
```

Exercise 2

- Sticker program is very useful.
- Write the sticker program
  - Can add to-do stickers
  - Can delete to-do sticker when it is finished

```
<title>What to do</title>
<link rel="stylesheet" href="middle1.css“ type="text/css">
<script>
    window.onload= start;
    function start(){
        for(var i = 0; i < localStorage.length; i++){
            var key = localStorage.key(i);
            var li = document.createElement("li");
            li.innerHTML = localStorage[key];
            var ul = document.getElementById("list");
            ul.appendChild(li);
        }
    }
    function addlist(){
        var input= document.getElementById("name");
        var add= document.createElement("li");
        add.innerHTML=input.value;
        var ul= document.getElementById("list");
        ul.appendChild(add);
        var str = "list"+localStorage.length;
        localStorage.setItem(str,input.value);
    }
</script>
<body>
```

```
    <form>
        <input type="text" id = "name">
        <input type="button" id = "addbutton" value ="스티커 노트 추가"
                                                onclick="addlist()">
        <p>
        <ul id= "list">
            <li>드라이클리닝 찾아오기</li>
            <li>케이블tv 취소하기</li>
        </ul>
    </form>
</body>
</html>
```

```
*{
   margin : 0.5em;
   padding : 0;
}
BODY {
   background : #acacac;        /*배경 회색*/
}
#name{
   width : 800px;
   height : 30px;
}
#addbutton{
   width : 150px;
   height : 30px;
   font-size : 15px;
   font-family: sans-serif;        }
#list ul{
   width : 1000px;
}
```

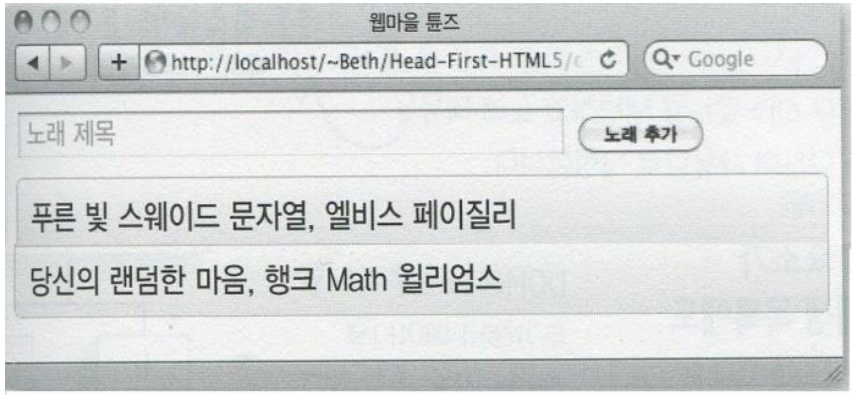
```
#list li{
   background : #ffffd9;        /*배경 노란색*/
   width : 400px;
   height : 300px;
   list-style: none;            /*li의 앞에 *제거*/
   float: left;
   font-size : 30px;
}
```

Web Storage

- Reference
  - http://wickedlysmart.com/hfhtml5/devtools.html
- Look up
  - notetoself4

Practice

- Calculator program
- Music manager program
  - Want to add in the list when button-on?




```
var li = document.createElement("li");

li.innerHTML = songName;

var ul = document.getElementById("playlist");

ul.appendChild(li);
```

Web worker

- JavaScript behavior
  - Single thread
  - Process several request/response sequentially
  - While processing request, a long time procedure would make the interaction with users longer
  - Slow script problem

  - Add multiple thread creation in HTML5: Web worker

Web worker

- Definition : http://www.w3.org/TR/workers/
- Working principle
  - Browser create one or more web workers for processing
  - Each worker is defined with its own JavaScript file including procedure description
  - Worker does not access runtime objects like variables, functions, and DOM of called pages
  - Browser sends messages for start worker
  - Worker send messages including work results
- Usage examples
  - Temporary storage for data
  - The spelling checker of input data
  - Popping up alert window in main page when an error occurs

```
<html>
<head>
<title>Ping Pong</title>
<meta charset="utf-8">
<script src="manager.js"></script>
</head>
<body>
<p id="output">
</p>
</body>
</html>
```

Manager.js

```
window.onload = function() {
        var worker = new Worker("worker.js");

        worker.postMessage("ping");
        worker.onmessage = function(event) {
                var msg= "Worker says " + event.data;
                alert(msg);
                document.getElementById("output").innerHTML =
                                                        msg;
        }
}
```

worker.js

```
onmessage = pingpong;

function pingpong(event) {
        if (event.data == "ping") {
                postMessage("pong");
        }
}
```

```
<!DOCTYPE HTML>
<html>
<head>
<title>Worker example: One-core computation</title>
</head>
<body>
  <p>The highest prime number discovered so far is:
  <output id="result"></output>
</p>
<script>
  var worker = new Worker('worker.js');
  worker.onmessage = function (event) {
          document.getElementById('result').textContent = event.data;
  };
</script>
</body> </html>
```

worker.js

```
var n = 1;
search: while (true) {
    n += 1;
    for (var i = 2; i <= Math.sqrt(n); i += 1)
        if (n % i == 0) continue search;
    // found a prime!
    postMessage(n);
}
```

Web worker

- postMessage
  - Supports for various message types

```
worker.postMessage("ping");
worker.postMessage([1, 2, 3]);
worker.postMessage({“message”: “ping”,
                    “count”: 5});

worker.postMessage(updateTheDom);
```

- Manager.js

Practice

- Write a program to check the id validity using web worker

Ajax

- Asynchronous Javascript and XML
- Based on Javascript and DOM
- New approach for server processing
- Asynchronous processing in the same page
- Use the ActiveXObject or XMLHttpRequest object instead of form submit

구매자 수	1012명
단가	250원
비용	80원

이익금 : **172,040**

판매수익 현황

AJAX-programming

- Step 1
  - Make the object to submit a request- request object
  - Use the built in ActiveXObject or XMLHttpRequest class
- Step 2
  - Write JavaScript function to set server connection and callback function
  - Use the request object made by step 1
- Step 3
  - Write the callback function to process the server response

```
<html>
 <head>
  <title>Boards 'R' Us</title>
  <link rel="stylesheet" type="text/css" href="boards.css" />
 </head>
 <body>
  <h1>Boards 'R' Us :: Custom Boards Report</h1>
  <div id="boards">
   <table>
    <tr><th>Snowboards Sold</th>
     <td><span id="boards-sold">1012</span></td></tr>
    <tr><th>What I Sell 'em For</th>
     <td>$<span id="price">249.95</span></td></tr>
    <tr><th>What it Costs Me</th>
     <td>$<span id="cost">84.22</span></td></tr>
   </table>
   <h2>Cash for the Slopes:
    $<span id="cash">167718.76</span></h2>
   <form method="GET" action="getUpdatedBoardSales.jsp">
    <input value="Show Me the Money" type="submit" />
   </form>
  </div>
 </body>
</html>
```

```
<%@ page language="“java” contentType="“text/html; charset=EUC-KR”
pageEncoding="“EUC-KR” %>
<%
// Start with an arbitrary number of boards sold
int totalSold = 1012;

// Reflect new sales
srand((double)microtime() * 1000000);
totalSold = totalSold + rand(0,1000);

double price = 249.95;
double cost = 84.22;
double cashPerBoard = price - cost;
double cash = totalSold * cashPerBoard;
%>

<html>
 <head>
  <title>Boards 'R' Us</title>
  <link rel="stylesheet" type="text/css" href="boards.css" />
 </head>
```

```
<body>
  <h1>Boards 'R' Us :: Custom Boards Report</h1>
  <div id="boards">
   <table>
    <tr><th>Snowboards Sold</th>
     <td><span id="boards-sold">
<%= totalSold %>
    </span></td></tr>
    <tr><th>What I Sell 'em For</th>
     <td>$<span id="price">
<%=price%>
    </span></td></tr>
    <tr><th>What it Costs Me</th>
     <td>$<span id="cost">
<%= cost %>
     </span></td></tr>   </table>
   <h2>Cash for the Slopes:
    $<span id="cash">
<%= $cash %>
</span></h2>
   <form method="GET"
action="getUpdatedBoardSales.jsp">
    <input value="Show Me the Money"
type="submit" />
   </form>
  </div>
 </body>
</html>
```

Step1

```
var request = null;

function createRequest() {
  try {
    request = new XMLHttpRequest();
  } catch (trymicrosoft) {
    try {
      request = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (othermicrosoft) {
      try {
        request = new ActiveXObject("Microsoft.XMLHTTP");
      } catch (failed) {
        request = null;
      }
    }
  }

  if (request == null)
    alert("Error creating request object!");
}
```

XMLHttpRequest class

- Methods
  - open(method, url[, async, username, passwd])
  - setRequestHeader(label, value)
  - send(content)
  - getAllResponseHeader()
  - getResponseHeader(label)
  - abort()

XMLHttpRequest class

- Properties
  - Onreadystatechange
  - readyState
    - Return the state of the request
    - Available value:

      0 request is uninitialized, 1 Connected

      2 started, 3 processed and send the respond

      4 ready to use the respond
  - responseText: text style response
  - ResponseXML: XML style response
  - status
    - Www.w3.org/Protocols/rfc2616//rfc2616-sec10.html
  - statusText: status info text

Step2

```
function getBoardsSold() {
    createRequest();
    var url = "getUpdatedBoardSales-ajax.jsp";
    request.open("GET", url, true);
    request.onreadystatechange = updatePage;
    request.send(null);
 }
```

Step 3-Server_side program

```
<%
// Start with an arbitrary number of boards sold
int totalSold = 1012;
// Reflect new sales
Random random = new Random();
totalSold = totalSold + random.nextInt(100);
out.println(totalSold);
%>
```

Step3– client_side collback functions

```
function updatePage() {
  if (request.readyState == 4) {
    var newTotal = request.responseText;
    var boardsSoldEl = document.getElementById("boards-sold");
    var cashEl = document.getElementById("cash");
    replaceText(boardsSoldEl, newTotal);

    /* Figure out how much cash Katie has made */
    var priceEl = document.getElementById("price");
    var price = getText(priceEl);
    var costEl = document.getElementById("cost");
    var cost = getText(costEl);
    var cashPerBoard = price - cost;
    var cash = cashPerBoard * newTotal;

    /* Update the cash for the slopes on the form */
    cash = Math.round(cash * 100) / 100;
    replaceText(cashEl, cash);
  }
}
```

```
function replaceText(el, text) {
  if (el != null) {
    clearText(el);
    var newNode = document.createTextNode(text);
    el.appendChild(newNode);
  }}
function clearText(el) {
  if (el != null) {
    if (el.childNodes) {
      for (var i = 0; i < el.childNodes.length; i++) {
        var childNode = el.childNodes[i];
        el.removeChild(childNode);
      }}}}
function getText(el) {
  var text = "";
  if (el != null) {
    if (el.childNodes) {
      for (var i = 0; i < el.childNodes.length; i++) {
        var childNode = el.childNodes[i];
        if (childNode.nodeValue != null) {
          text = text + childNode.nodeValue;
        }}}}
  return text;
}
```

Sample 2

```
<!DOCTYPE html >
<html><head><title>Hello Ajax World</title>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR" />
<style type="text/css">
div.elem { margin: 20px; }
</style>
<script type="text/javascript">
var xmlhttp = false;
if (window.XMLHttpRequest) {
   xmlhttp = new XMLHttpRequest( );
   xmlhttp.overrideMimeType('text/xml');
} else if (window.ActiveXObject) {
   xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
function populateList( ) {
   var state = document.forms[0].elements[0].value;
   var url = 'ajax.jsp?state=' + state;
   xmlhttp.open('GET', url, true);
   xmlhttp.onreadystatechange = getCities;
   xmlhttp.send(null);

}
```

- How can we use post methods for request submit?

For post method

```
function populateList(  ) {
   var state = document.forms[0].elements[0].value;
   var qry = "state="+ state;
   var url = 'ajax.jsp?';
   xmlhttp.open( 'POST', url, true);
   xmlhttp.onreadystatechange = getCities;
   xmlhttp.setRequestHeader("Content-type", "application/x-www-form-
     urlencoded");
   xmlhttp.send(qry);

}
```