Introduction to Web Technology

HTML5: Client-Side Storage

Joseph Tonien
School of Computing and Information Technology
University of Wollongong

Client-Side Web Storage

- Store data on the client side, instead of the server
- Make the web application available offline
- The storage is per origin (protocol + domain + port)
- Simple storage: data is stored in name/value pair

2 types of storage:

- **localStorage**: a single persistent object which stores data with no expiration date;
- **sessionStorage**: stores data for one session only, data is cleared when the browser tab is closed.

Client-Side Web Storage

Checking if the browser supports web storage or not:

```
// return true if local storage is supported
// otherwise return false
function storageSupported(){
  if (typeof(Storage) !== "undefined") {
    return true;
  } else {
    return false;
  }
}
```

Client-Side Web Storage

Storing and retrieving data from Web Storage:

```
// storing data to the localStorage
localStorage.setItem("the-key", "the-value");

// get data from localStorage
var the-value = localStorage.getItem("the-key");
```

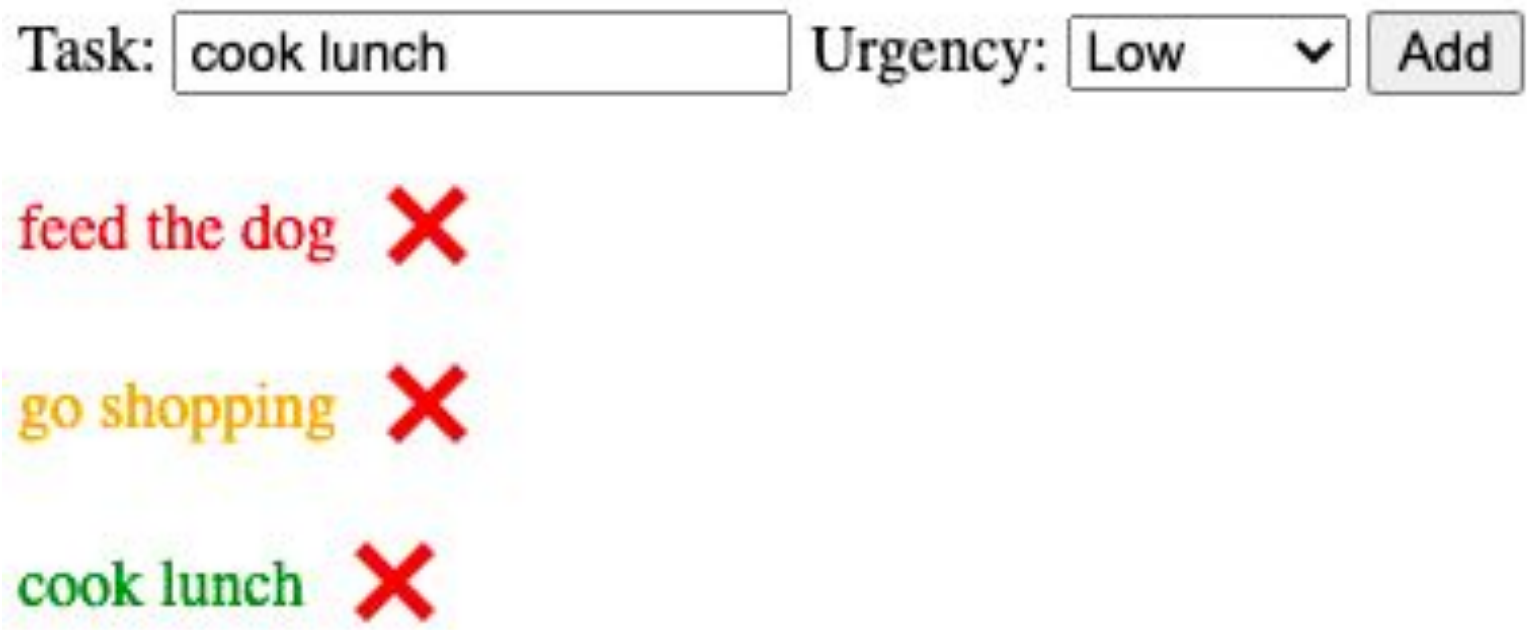
Removing data from Web Storage:

```
// removing data to the localStorage
localStorage.removeItem("the-key");
```

Example: To-Do-List

We want to create a web application where the user can create a to-do-list and save it to the local storage.

We will store the JSON of the task list into the local storage.



Example: To-Do-List

design the HTML elements

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

a text field for user to enter task description

```
<input type="text" id="task" />
```

Example: To-Do-List

design the HTML elements

Task: cook lunch Urgency: Low Add

feed the dog

go shopping

cook lunch

a selection for user to select task urgency

```
<select id="urgency">
  <option value="High">High</option>
  <option value="Medium">Medium</option>
  <option value="Low" selected="selected">Low</option>
</select>
```

Example: To-Do-List

design the HTML elements

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

button to add task

```
<button onClick="addTask()">
  Add
</button>
```

Example: To-Do-List

design the HTML elements

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

a div to display all the tasks

```
<div id="taskDisplay">
</div>
```

Example: To-Do-List

design the data structure

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

```
{
  id:12345xxx...,
  task:"feed the dog",
  urgency:"High"
}
```

```
{
  id:yyy...,
  task:"go shopping",
  urgency:"Medium"
}
```

each task is an object

```
{
  id:zzz...,
  task:"cook lunch",
  urgency:"Low"
}
```

Example: To-Do-List

design the data structure

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

store all tasks into an array variable

```
[
  {
    id:12345xxx...,
    task:"feed the dog",
    urgency:"High"
  },
  {
    id:yyy...,
    task:"go shopping",
    urgency:"Medium"
  },
  {
    id:zzz...,
    task:"cook lunch",
    urgency:"Low"
  }
]
```

Example: To-Do-List

design the data structure

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

store all tasks into an array variable

```
// list of tasks
// each task is an object that contains:
// {
//   id: the task id (the time when task created)
//   task: the task name
//   urgency: the task urgency (High, Medium, or Low)
// }
var toDoList = [];
```

Example: To-Do-List

design the local storage

Task: Urgency: Low Add

translate task array into JSON string

toDoList → toDoListJSON

```
[
  {
    id:12345xxx...,
    task:"feed the dog",
    urgency:"High"
  },
  {
    id:yyy...,
    task:"go shopping",
    urgency:"Medium"
  },
  {
    id:zzz...,
    task:"cook lunch",
    urgency:"Low"
  }
]
```

```
[
  {
    "id":12345xxx...,
    "task":"feed the dog",
    "urgency":"High"
  },
  {
    "id":yyy...,
    "task":"go shopping",
    "urgency":"Medium"
  },
  {
    "id":zzz...,
    "task":"cook lunch",
    "urgency":"Low"
  }
]
```

Example: To-Do-List

design the local storage

Task: Urgency: Low Add

store JSON string into local storage

```
toDoListJSON
```

```
[
  {
    "id":12345xxx...,
    "task":"feed the dog",
    "urgency":"High"
  },
  {
    "id":yyy...,
    "task":"go shopping",
    "urgency":"Medium"
  },
  {
    "id":zzz...,
    "task":"cook lunch",
    "urgency":"Low"
  }
]
```

Key	Value
toDoListJSON	[{"id":1605572931427, "task":"feed the dog", "urgency":"High"},...]

Example: To-Do-List

Function: Add a task

Task: feed the dog Urgency: High Add

```
get task info from user and create task object
```

```
toDoList
add task object to the list of tasks
```

```
<div id="taskDisplay">
</div>
```

```
display updated list of tasks
```

```
store updated list of tasks into local storage
```

Key	Value
toDoListJSON	[{"id":1605572931427, "task":"feed the dog", "urgency":"High"},...]

Example: To-Do-List

Function: Add a task


```
function addTask(){
  // get task info from user and create task object
  var taskObj = createTask();

  // add task object to the list of tasks
  toDoList.push(taskObj);

  // display updated list of tasks
  displayTasks();

  // store updated list of tasks into local storage
  saveTasksToLocal();
}
```

Example: To-Do-List

Function: Add a task


```
function createTask(){
  // get task info from user
  var taskTf = document.getElementById( "task");
  var taskName = taskTf.value;

  var urgencySelect = document.getElementById( "urgency");
  var taskUrgency = urgencySelect.value;

  // create task object
  var taskObj = {};
  var currentDate = new Date();
  taskObj.id = currentDate.getTime();
  taskObj.task = taskName;
  taskObj.urgency = taskUrgency;

  // return task object
  return taskObj;
}
```

Example: To-Do-List

Function: Display tasks

Task: feed the dog Urgency: High Add

```
function displayTasks(){
  // construct the html contains all the tasks
  var html = "";

  // use for loop to go through all the tasks
  for(var i=0; i < toDoList.length; i++){
    var taskObj = toDoList[i];
    var taskHTML = getTaskHTML(taskObj);
    html = html + taskHTML;
  }

  // display tasks in the DIV
  var displayDiv = document.getElementById( "taskDisplay");
  displayDiv.innerHTML = html;
}
```

Example: To-Do-List

Function: Display tasks

Task: feed the dog Urgency: High Add

feed the dog ✕

```
function getTaskHTML(taskObj){
  // construct the html for displaying the task
  var html = "<p>";

  // task description in color
  var taskDesc = getTaskDescriptionHTML(taskObj);
  html += taskDesc;

  // task button
  var taskButton = getTaskDeleteButtonHTML(taskObj);
  html += taskButton;

  html += "</p>";

  return html;
}
```

Example: To-Do-List

Function: Display tasks

Task: feed the dog Urgency: High Add

feed the dog ✕

```
function getTaskDescriptionHTML(taskObj){
  // using different color for the urgency
  var desc = "";
  if (taskObj.urgency == "High"){
    desc = "<span style='color:red;'>"
           + taskObj.task
           + "</span>";
  }
  else if (taskObj.urgency == "Medium"){
    desc = "<span style='color:orange;'>"
           + taskObj.task
           + "</span>";
  }
  else if (taskObj.urgency == "Low"){
    desc = "<span style='color:green;'>"
           + taskObj.task
           + "</span>";
  }
  return desc;
}
```

Example: To-Do-List

Function: Display tasks

Task: feed the dog Urgency: High Add

feed the dog ✕

```
function getTaskDeleteButtonHTML(taskObj){
  var deleteEmoji = "&#10060;";

  var deleteButton = "<span onClick='deleteTask("  + taskObj.id + ")'>"
                     + deleteEmoji
                     +  "</span>";

  return deleteButton;
}

function deleteTask(taskId){

  ...

}
```

Example: To-Do-List

Function: Delete a task

Task: feed the dog Urgency: High Add

feed the dog ✕

```
toDoList
remove task object from the list of tasks
```

```
<div id="taskDisplay">
</div>
```

```
display updated list of tasks
```

```
store updated list of tasks into local storage
```

Key	Value
toDoListJSON	[{"id":1605572931624, "task":"cook lunch", "urgency":"Low"},...]

Example: To-Do-List

Function: Delete a task

Task: feed the dog Urgency: High Add

feed the dog ✕

```
function deleteTask(taskId){
  // remove task object from the list of tasks
  // search for task id
  for(var i=0; i < toDoList.length; i++){
    var taskObj = toDoList[i];
    if (taskObj.id == taskId){
      // delete task out of the task array
      toDoList.splice(i, 1);
    }
  }

  // display updated list of tasks
  displayTasks();

  // store updated list of tasks into local storage
  saveTasksToLocal();
}
```

Example: To-Do-List

Function: save tasks to local storage

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

```
toDoList
   |
   |   translate task array into JSON string
   v
toDoListJSON
   |
   |   store JSON string into local storage
   v
```

Key	Value
toDoListJSON	[{"id":1605572931235, "task":"feed the dog", "urgency":"High"},...]

Example: To-Do-List

Function: save tasks to local storage

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

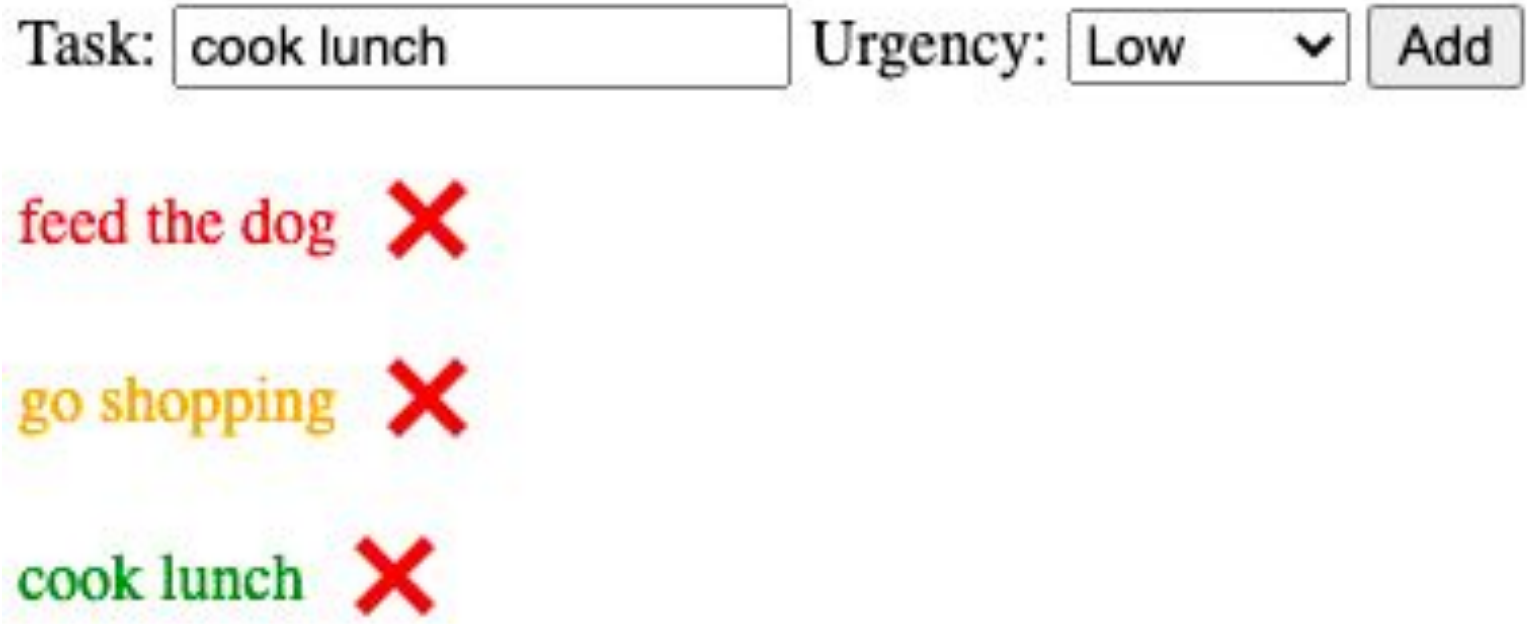
cook lunch ✕

```
function saveTasksToLocal(){
  // check if local storage supported
  if(storageSupported()){
    // translate task array into JSON string
    var toDoListJSON = JSON.stringify(toDoList);

    // store JSON string into local storage
    localStorage.setItem("toDoListJSON", toDoListJSON);
  }
}
```

Example: To-Do-List

Function: save tasks to local storage




```
// return true if local storage is supported
// otherwise return false
function storageSupported(){
  if (typeof(Storage) !== "undefined") {
    return true;
  } else {
    return false;
  }
}
```

Example: To-Do-List

Function: load tasks

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

What happen when the user closes the website and then comes back on another day?

Example: To-Do-List

Function: load tasks

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

What happen when the user closes the website and then comes back on another day?

When the website loaded, we need to read the local storage for the saved list of tasks and then we need to display this saved list of tasks.

Example: To-Do-List

Function: load tasks

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

Key	Value
toDoListJSON	[{"id":1605572931235, "task":"feed the dog", "urgency":"High"},...]

read saved JSON from local storage

toDoListJSON

translate JSON string to task array

toDoList

display list of tasks

```
<div id="taskDisplay">
</div>
```

Example: To-Do-List

Function: load tasks

Task: cook lunch Urgency: Low Add

feed the dog ✕

go shopping ✕

cook lunch ✕

```
<body onLoad="loadTasks()">
```

```
function loadTasks(){
  // check if local storage supported
  if(storageSupported()){
    // read saved JSON from local storage
    var toDoListJSON = localStorage.getItem( "toDoListJSON");

    // translate JSON string to task array
    if(toDoListJSON != null){
      toDoList = JSON.parse(toDoListJSON);
    }

    // display list of tasks
    displayTasks();
  }
}
```

References

- https://www.w3.org/TR/webstorage/
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API