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:

```
if (typeof(Storage) !== "undefined") {
    // code for web storage ...
} else {
    console.log("Web Storage not supported");
}
```

Client-Side Web Storage

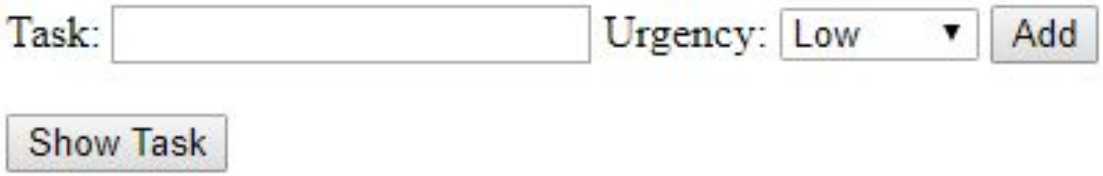
Storing and retrieving data from Web Storage:

```
// 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 (1)



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:

Key	Value
toDoListJSON	[{"task":"Grocery shopping","urgency": "Low"},...]

```
[
  {
    "task":"Grocery shopping",
    "urgency":"Low"
  },
  {
    "task":"Wash the dishes",
    "urgency":"Medium"
  },
  {
    "task":"Do web tech assignment 2",
    "urgency":"High"
  },
  {
    "task":"Do laundry",
    "urgency":"Low"
  }
]
```

Example: To-Do-List (1)

When the page load, we need to do the database initialization:
(i) get the to-do list JSON from the local storage,
(ii) parse the JSON.

```
<body onLoad="initApp();">
…
// initialize the application
function initApp(){
  if (typeof(Storage) !== "undefined"){

    // get the to-do list JSON from local storage

    // parse the JSON to the toDoList

  } else{
    console.log("Web Storage not supported");
  }
}
```

Example: To-Do-List (1)

```
// to-do list which is saved to web storage
var toDoList = [];

function initApp(){
  if (typeof(Storage) !== "undefined"){

    // get the to-do list JSON from local storage
    var toDoListJSON = localStorage.getItem("toDoListJSON");

    // parse the JSON to the toDoList
    if(toDoListJSON != null){
      toDoList = JSON.parse(toDoListJSON);
    }

  } else{
    console.log("Web Storage not supported");
  }
}
```

Example: To-Do-List (1)

Adding a task to the to-do-list:

```
<body onLoad="initApp();">

Task:
<input id="task" type="text" />

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

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

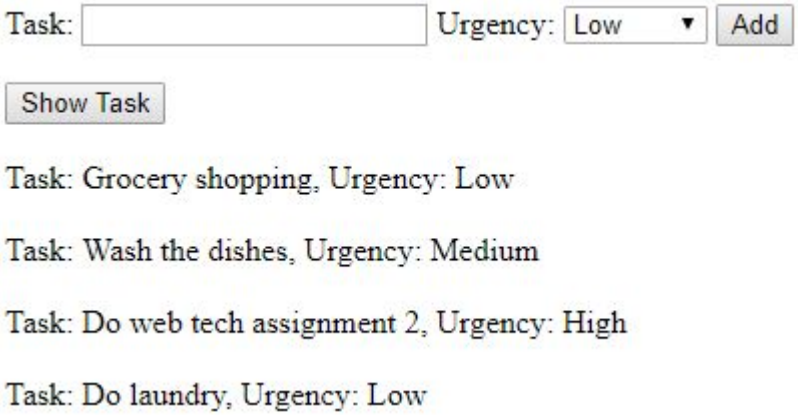
Example: To-Do-List (1)

Adding a task to the to-do-list:

```
// add a task
function addTask(){
  // get task description from user input
  var toDoObj = {};
  toDoObj.task = document.getElementById("task").value;
  toDoObj.urgency = document.getElementById("urgency").value;

  // add the task to toDoList
  toDoList.push(toDoObj);


  // if Web Storage supported then update the JSON
  if (typeof(Storage) !== "undefined"){
    localStorage.setItem("toDoListJSON", JSON.stringify(toDoList) );
  }
}
```

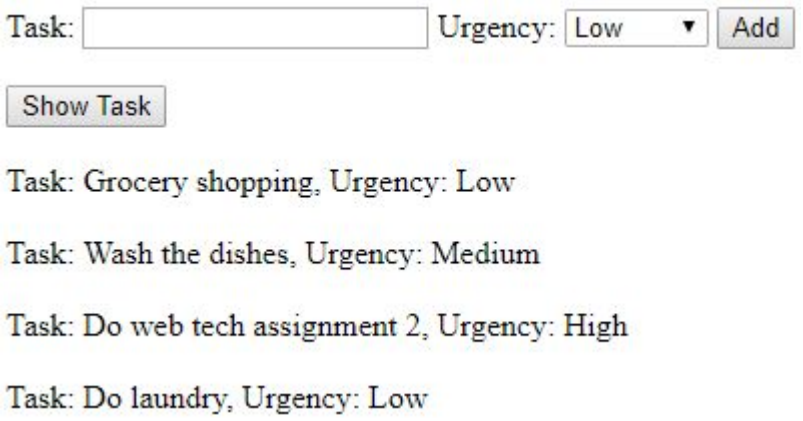
Example: To-Do-List (1)

Showing all the tasks:




```
<button onClick="showTask()">
Show Task
</button>

<br /><br />

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

Example: To-Do-List (1)



Showing all the tasks:

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

```
// show all the tasks
function showTask(){
  var html = "";

  for (var i=0; i<toDoList.length; i++) {
    var toDo = toDoList[i];

    html += "Task: " + toDo.task + ", Urgency: " + toDo.urgency + "<br /><br />";
  }

  document.getElementById("taskDisplay").innerHTML = html;
}
```

Example: To-Do-List (2)

To-Do-List(2) example is the same as the previous To-Do-List(1) example, except that each task is displayed with a color corresponding to its urgency level.

Example: To-Do-List (2)


```
// show all the tasks
function showTask(){
  var html = "";

  for (var i=0; i<toDoList.length; i++) {
    var toDo = toDoList[i];

    if(toDo.urgency == "Low"){
      html += "<span style='color:green'>" + toDo.task + "</span>";
    }else if(toDo.urgency == "Medium"){
      html += "<span style='color:orange'>" + toDo.task + "</span>";
    }else if(toDo.urgency == "High"){
      html += "<span style='color:red'>" + toDo.task + "</span>";
    }

    html += "<br /><br />";
  }

  document.getElementById("taskDisplay").innerHTML = html;
}
```

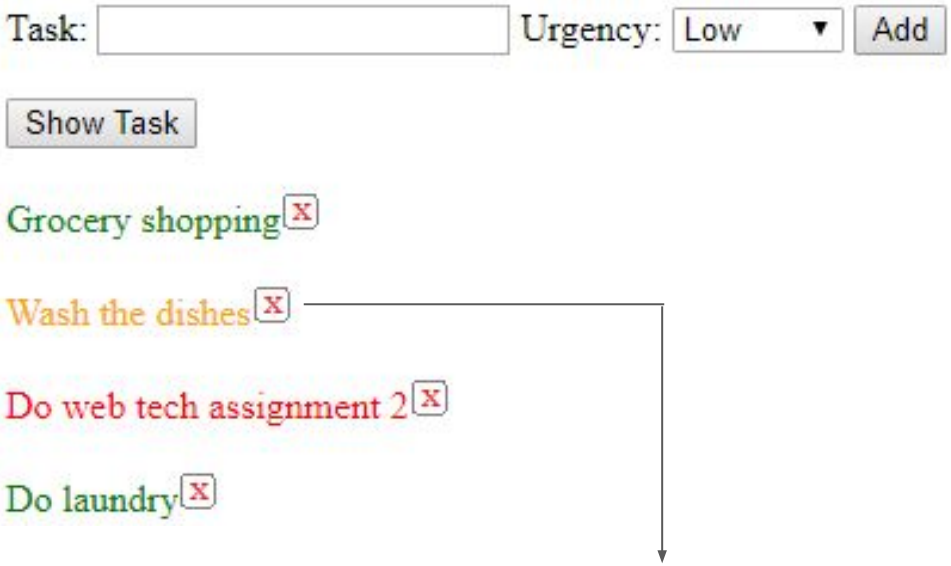
Example: To-Do-List (3)

To-Do-List(3) example is the same as the previous To-Do-List(2) example, except that each task is displayed with a delete symbol, and when the user clicks on the delete symbol the task will be deleted.

We will use an image `delete.png` for the delete symbol.

Example: To-Do-List (3)

We will use an image `delete.png` for the delete symbol.

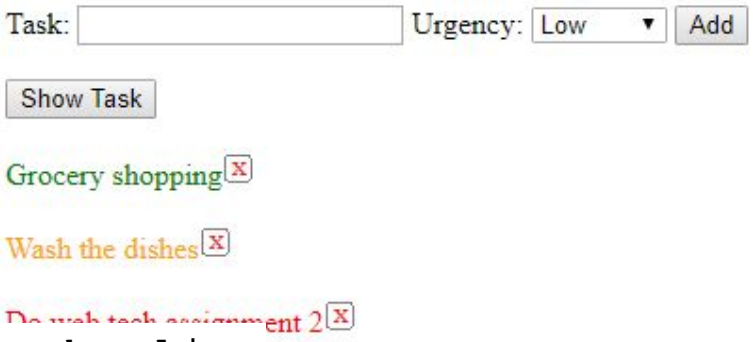



```
<img src='delete.png' onClick='deleteTask("Wash the dishes")'/>
```

Example: To-Do-List (3)


```
// show all the tasks
function showTask(){
  var html = "";

  for (var i=0; i<toDoList.length; i++) {
    var toDo = toDoList[i];

    if(toDo.urgency == "Low"){
      html += "<span style='color:green'>" + toDo.task + "</span>";
    }else if(toDo.urgency == "Medium"){
      html += "<span style='color:orange'>" + toDo.task + "</span>";
    }else if(toDo.urgency == "High"){
      html += "<span style='color:red'>" + toDo.task + "</span>";
    }

    html += "<img src='delete.png' onClick=' deleteTask(\"" + toDo.task + "\")'/>";
    html += "<br /><br />";
  }

  document.getElementById("taskDisplay").innerHTML = html;
}
```

Example: To-Do-List (3)

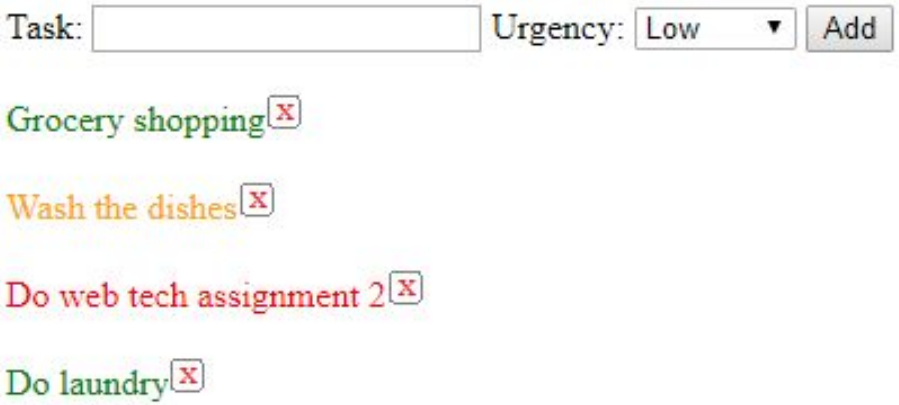



```
// delete a task
function deleteTask(task){

  // search for the deleted task through the list
  for (var i=0; i<toDoList.length; i++) {
    var toDo = toDoList[i];

    if(toDo.task == task){
      toDoList.splice(i, 1);
      break;
    }
  }

  // if Web Storage supported then update the JSON
  if (typeof(Storage) !== "undefined"){
    localStorage.setItem("toDoListJSON", JSON.stringify(toDoList) );
  }

  // show all the tasks
  showTask();
}
```

Example: To-Do-List (4)

To-Do-List(4) example is the same as the previous To-Do-List(3) example, except that the button “Show Task” is removed. Initially, all tasks will be displayed and after adding a new task, the list of updated tasks will be displayed.

Example: To-Do-List (4)




```
<body onLoad="initApp();">
…
// initialize the application
function initApp(){
  if (typeof(Storage) !== "undefined"){

    // get the to-do list JSON from local storage

    // parse the JSON to the toDoList

    // show all the tasks

  } else{
    console.log("Web Storage not supported");
  }
}
```

Example: To-Do-List (4)

```
// to-do list which is saved to web storage
var toDoList = [];

function initApp(){
  if (typeof(Storage) !== "undefined"){

    // get the to-do list JSON from local storage
    var toDoListJSON = localStorage.getItem("toDoListJSON");

    // parse the JSON to the toDoList
    if(toDoListJSON != null){
      toDoList = JSON.parse(toDoListJSON);
    }

    // show all the tasks
    showTask();

  } else{
    console.log("Web Storage not supported");
  }
}
```

Example: To-Do-List (4)

Adding a task to the to-do-list:

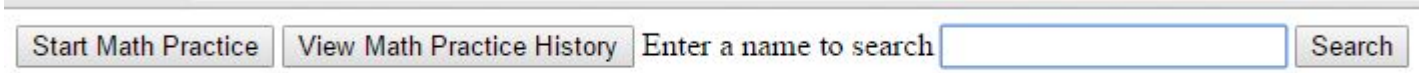

```
// add a task
function addTask(){
  // get task description from user input
  var toDoObj = {};
  toDoObj.task = document.getElementById("task").value;
  toDoObj.urgency = document.getElementById("urgency").value;

  // add the task to toDoList
  toDoList.push(toDoObj);


  // if Web Storage supported then update the JSON
  if (typeof(Storage) !== "undefined"){
    localStorage.setItem("toDoListJSON", JSON.stringify(toDoList) );
  }

  // show all the tasks
  showTask();
}
```

Example: To-Do-List (4)

```
// delete a task
function deleteTask(task){

  // search for the deleted task through the list
  for (var i=0; i<toDoList.length; i++) {
    var toDo = toDoList[i];

    if(toDo.task == task){
      toDoList.splice(i, 1);
      break;
    }
  }

  // if Web Storage supported then update the JSON
  if (typeof(Storage) !== "undefined"){
    localStorage.setItem("toDoListJSON", JSON.stringify(toDoList) );
  }

  // show all the tasks
  showTask();
}
```

Example: Math Practice

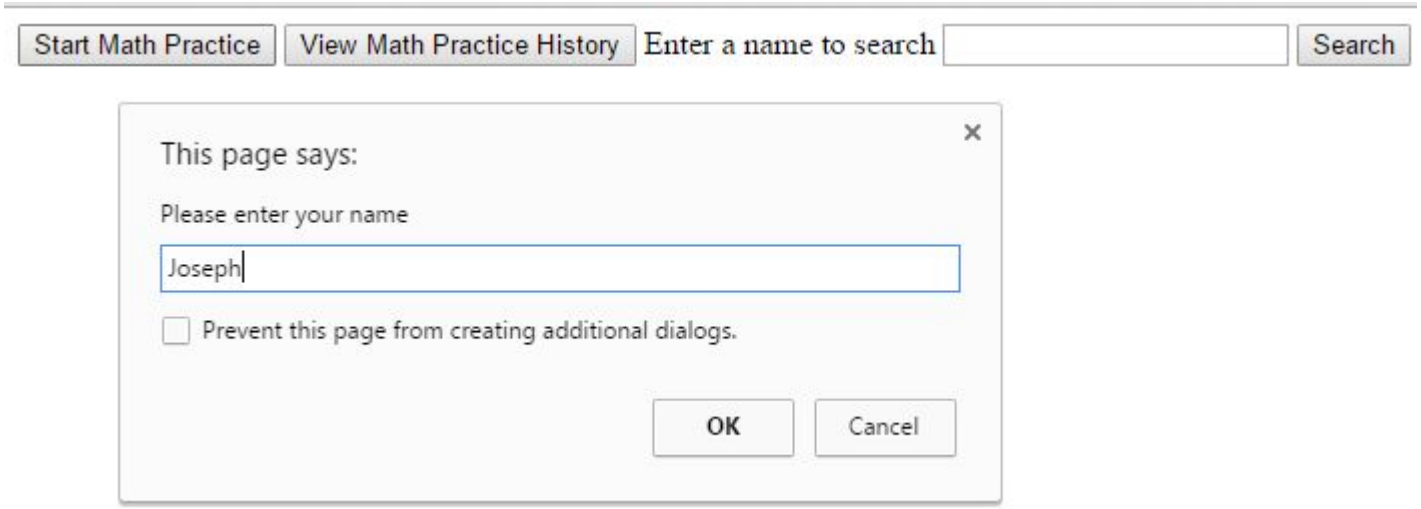
We want to create a web application where children can practice mathematics and parents can view the result of their kids practice.



Example: Math Practice

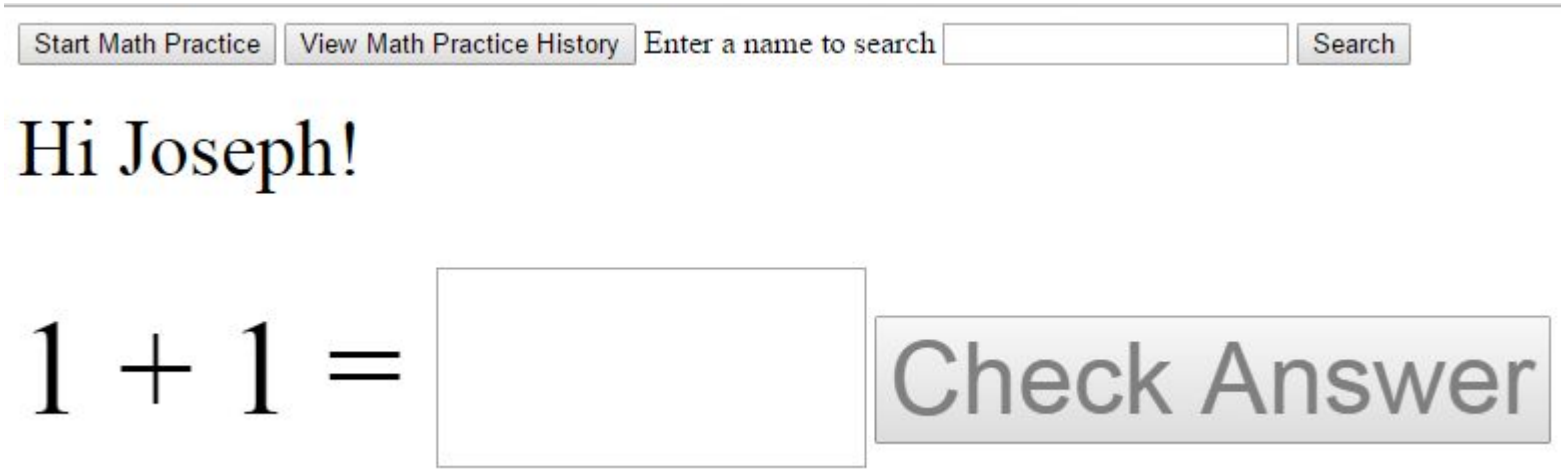
This is how the application should work.

When the user clicks on the button “Start Math Practice”, the user will be asked to enter his/her name.



Example: Math Practice

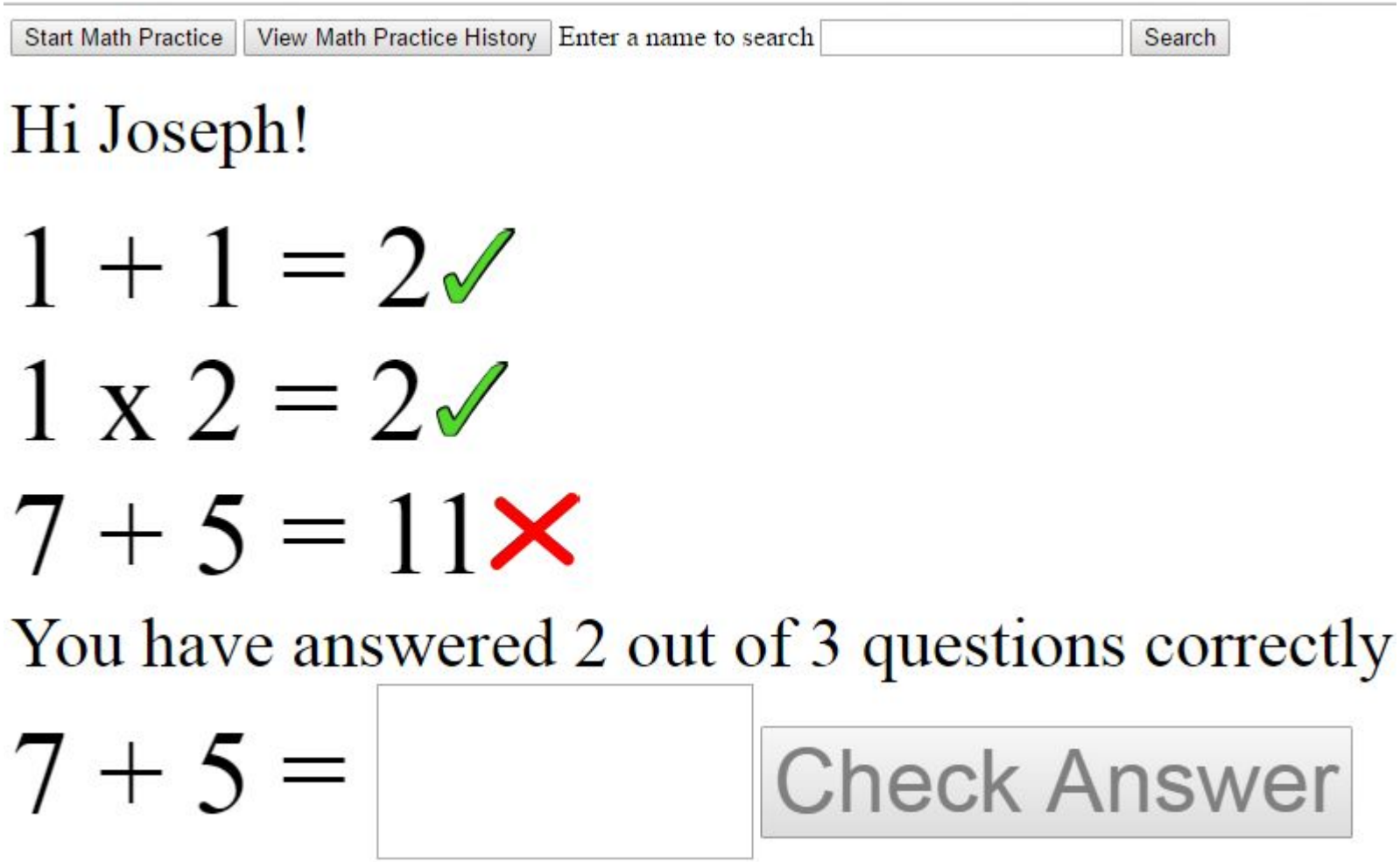
Then the application prints a greetings and generate a math question.



Example: Math Practice

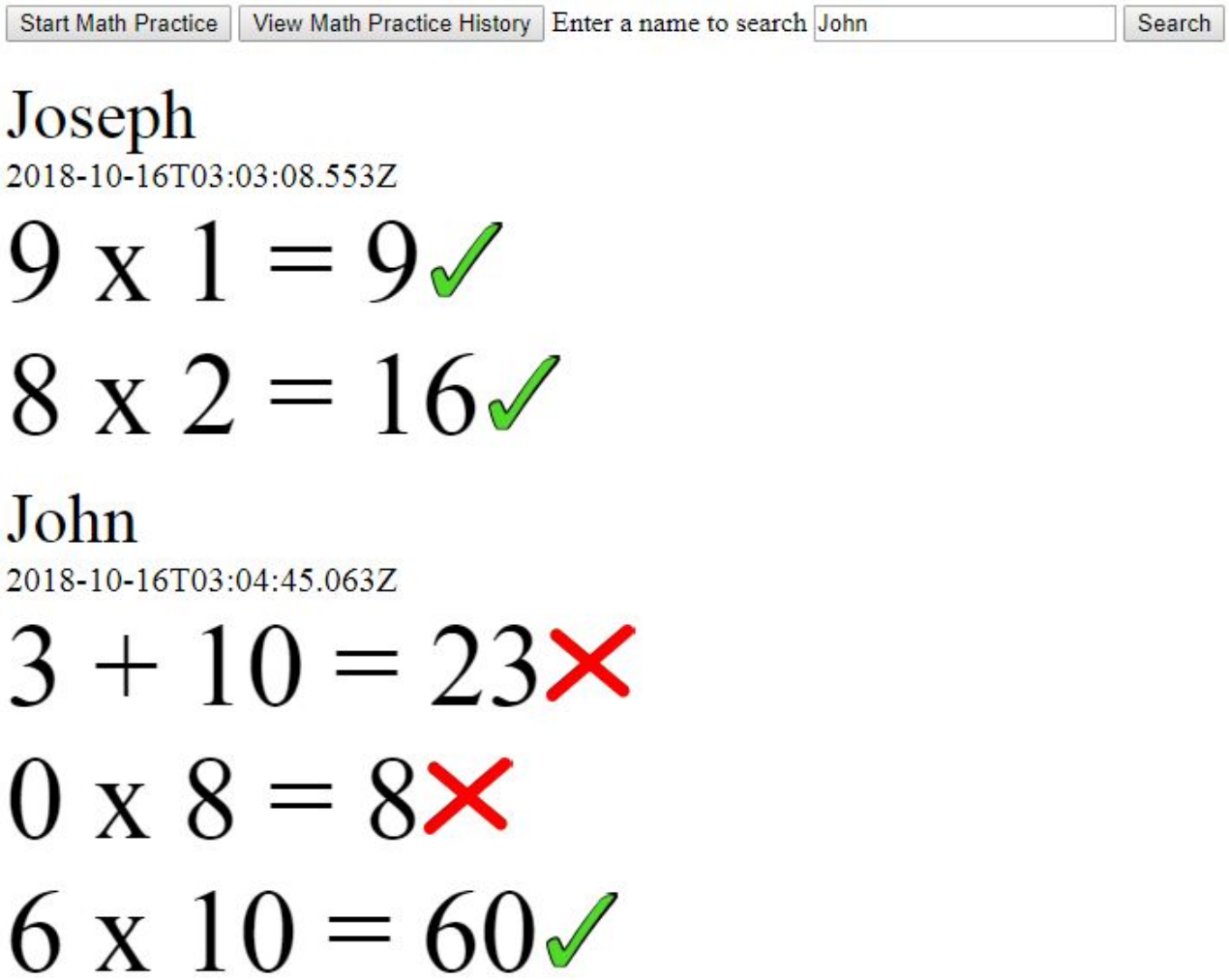
User can enter an answer to the math problem and can check if it is correct.

New question will be generated.



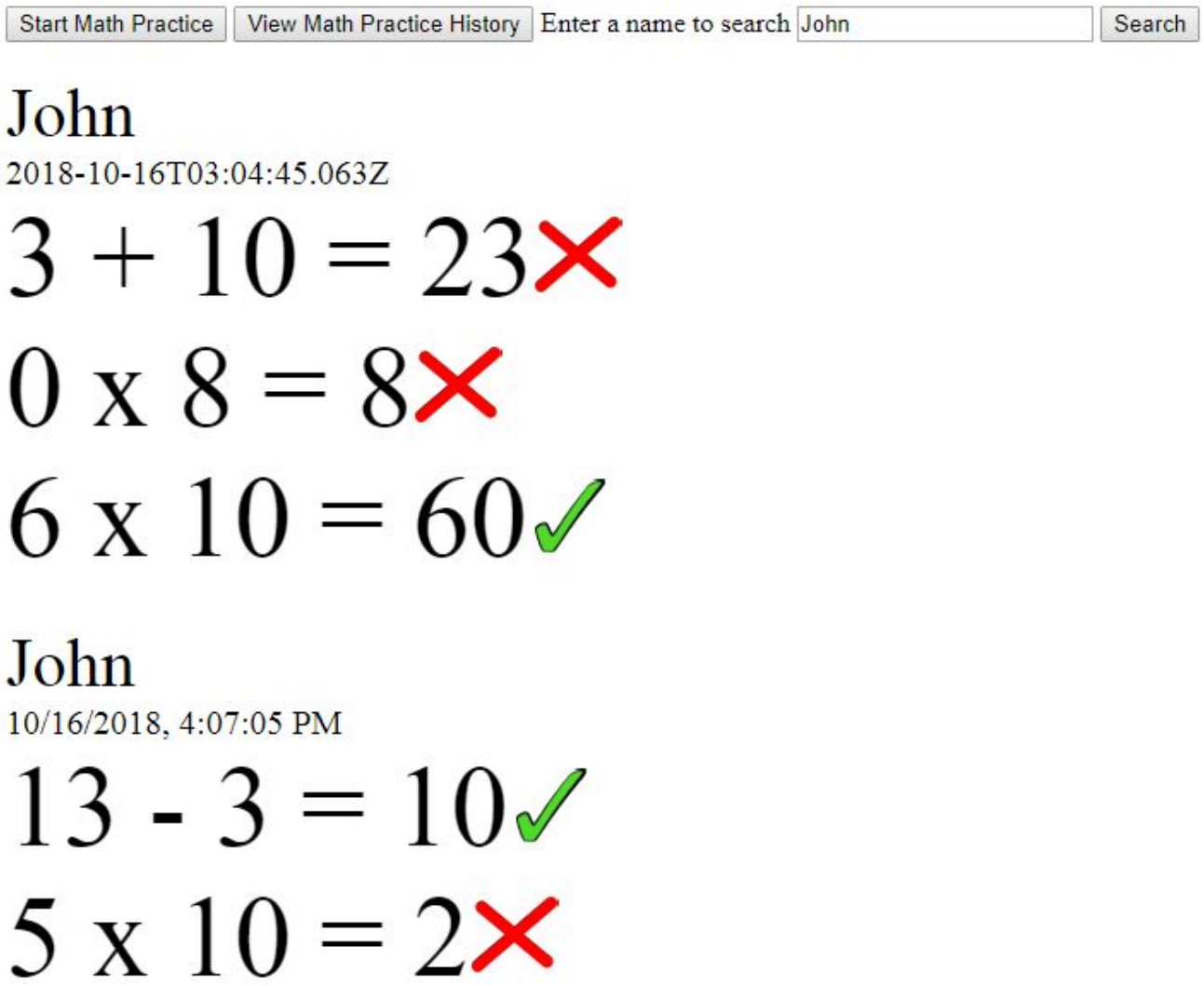
Example: Math Practice

Parents can click on the button “View Math Practice History” to see the result of their kids practice.



Example: Math Practice

Parents can enter a name to search



References

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