CSIT884:
Web Development

Local Storage (Web Storage API)

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 key value (or 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:

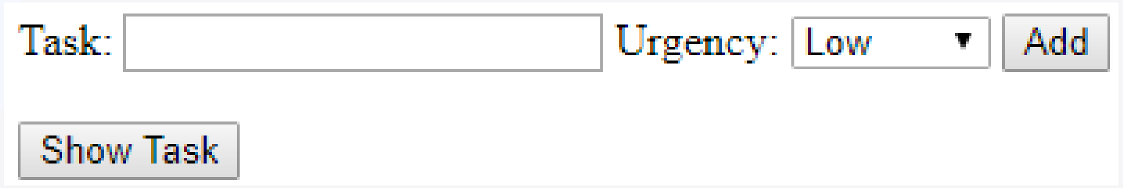
```
// storing data to the localStorage
localStorage.setItem("the-key", "the-value"); . . . . (1)

// get data from localStorage
var theValue = localStorage.getItem("the-key"); . . . (2)
```

Client-Side Web Storage

Removing data from Web Storage:

```
// removing data to the localStorage
localStorage.removeItem("the-key"); . . . . (1)

// remove all data of the localStorage for the specific domain
localStorage.clear(); . . . . (2)
```

Example: To-Do-List (1)



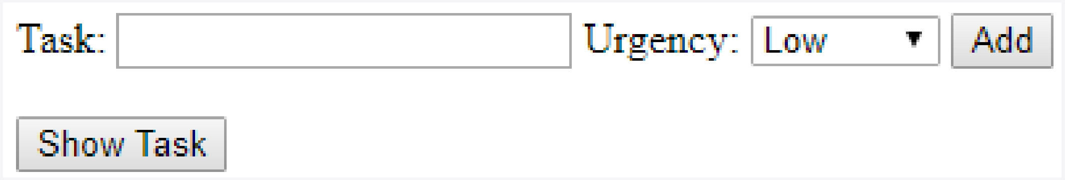
We want to create a web application where the user can create a to-do-list and save it to the local storage.

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

Key	Value
toDoListJSON	[{"time":1605572931427, "task":"Grocery shopping", "urgency":"Low"},...]

```
[
  {
    "time":1605572931427,
    "task":"Grocery shopping",
    "urgency":"Low"
  },
  {
    "time":1605573037767,
    "task":"Wash the dishes",
    "urgency":"Medium"
  },
  {
    "time":1605573073185,
    "task":"Do laundry",
    "urgency":"Low"
  }
]
```

Example: To-Do-List (1)



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

. . .
```

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.

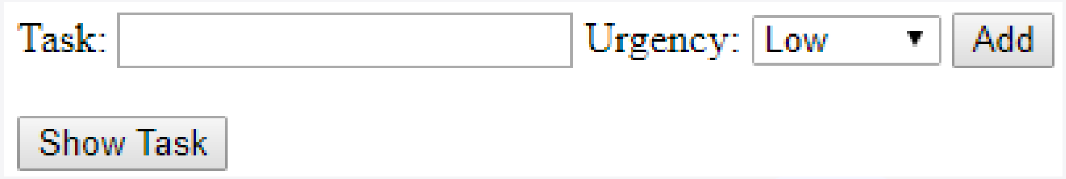
```
<body onLoad="initApp();">
…
// initialize the application
function initApp(){
  if (storageSupported()){

    // 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 = []; . . . .(1)
function initApp(){
  if (storageSupported()){

    // get the to-do list JSON from local storage
    var toDoListJSON = localStorage.getItem("toDoListJSON");..(2)

    // parse the JSON to the toDoList
    if(toDoListJSON != null){ . . . . . . . . . (3)
      toDoList = JSON.parse(toDoListJSON); . . . . (4)
    }

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

Example: To-Do-List (1)

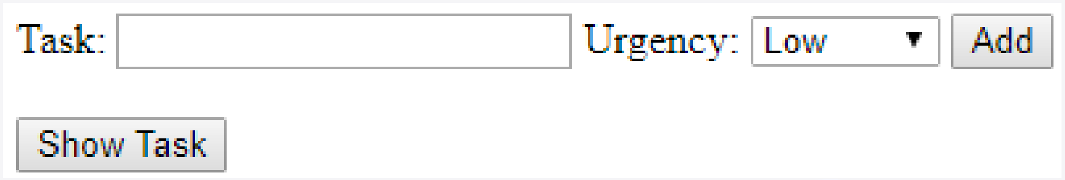
Adding a task to the to-do-list:



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


. . .
```

Example: To-Do-List (1)

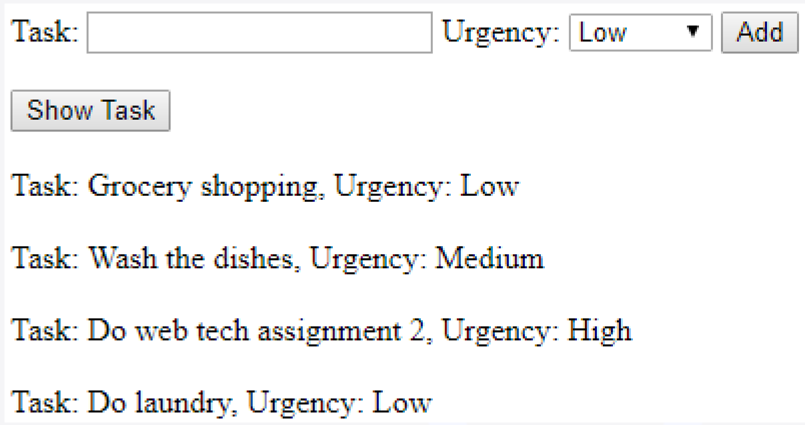
Adding a task to the to-do-list:



```
// add a task
function addTask(){
  // get task description from user input
  var toDoObj = {};

  var now = new Date();
  toDoObj.time = now.getTime();
  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 (storageSupported()){
    localStorage.setItem("toDoListJSON", JSON.stringify(toDoList));
  }
}
```

Example: To-Do-List (1)

Showing all the tasks:



Task: Grocery shopping, Urgency: Low

Task: Wash the dishes, Urgency: Medium

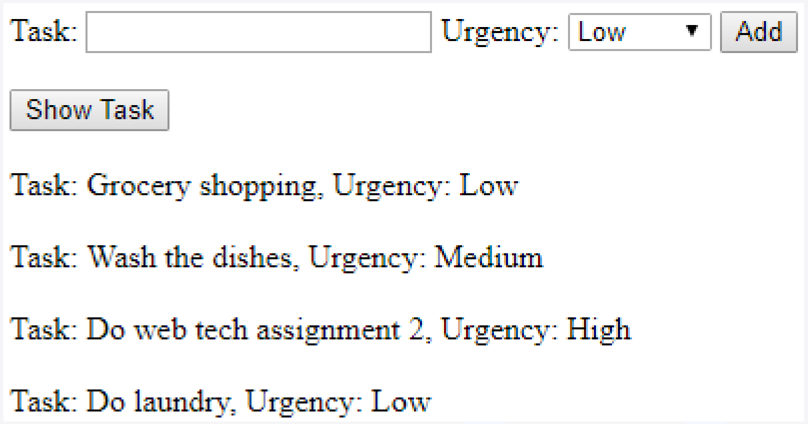
Task: Do web tech assignment 2, Urgency: High

Task: Do laundry, Urgency: Low

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

<br /><br />

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

Example: To-Do-List (1)

Showing all the tasks:



Task: Grocery shopping, Urgency: Low

Task: Wash the dishes, Urgency: Medium

Task: Do web tech assignment 2, Urgency: High

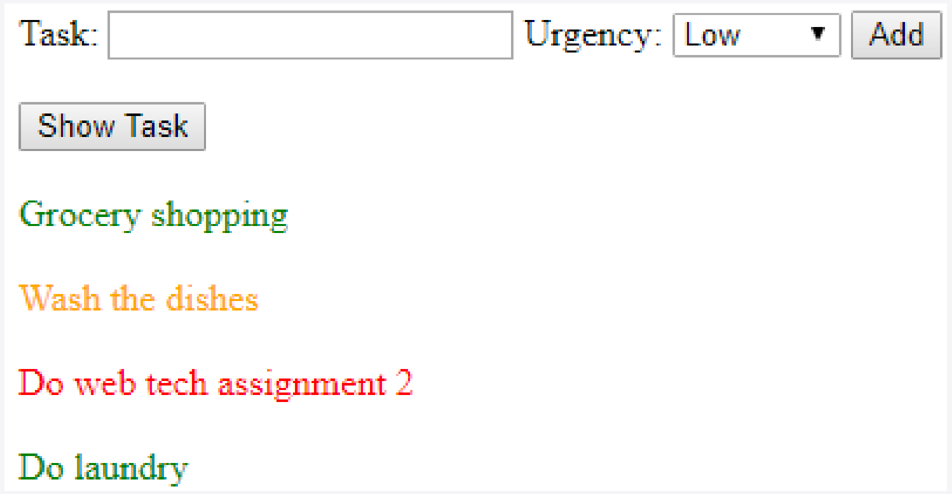
Task: Do laundry, Urgency: Low

```
<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.



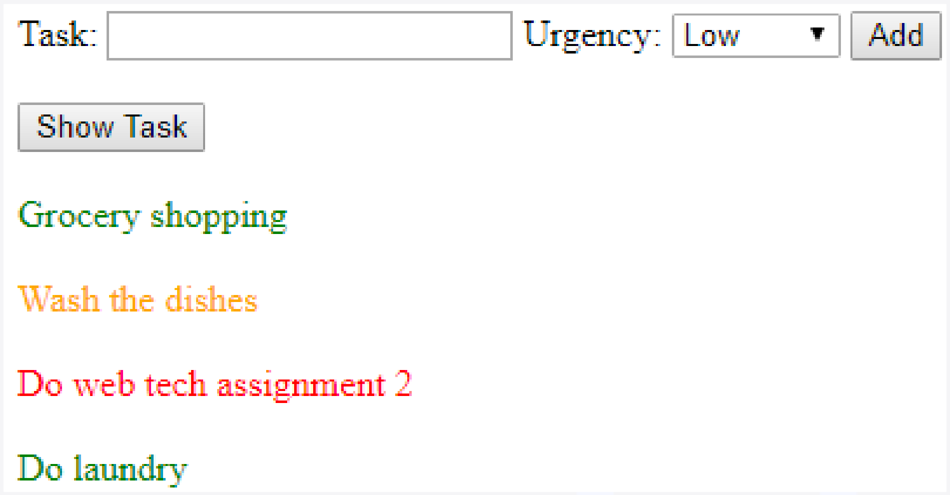
Grocery shopping

Wash the dishes

Do web tech assignment 2

Do laundry

Example: To-Do-List (2)



Grocery shopping

Wash the dishes

Do web tech assignment 2

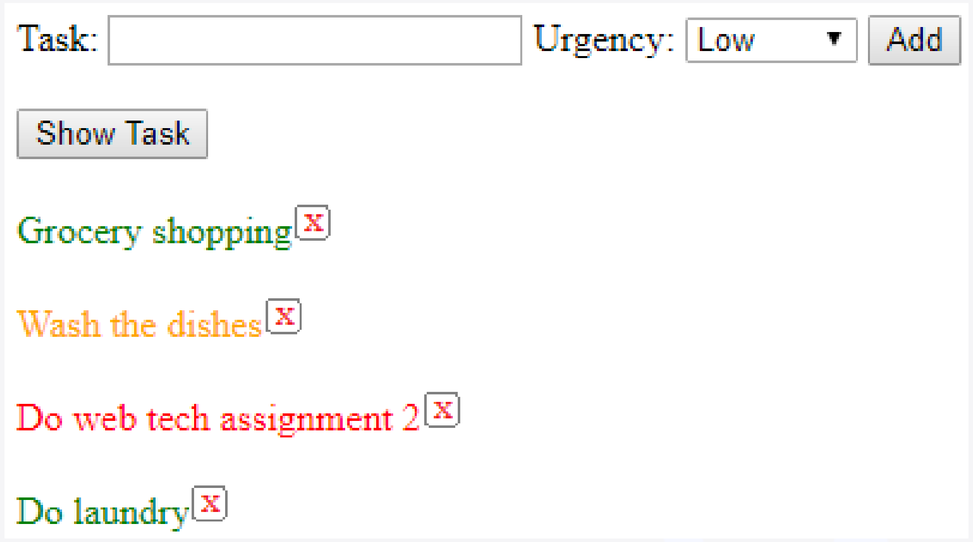
Do laundry

```
// 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.



Grocery shopping

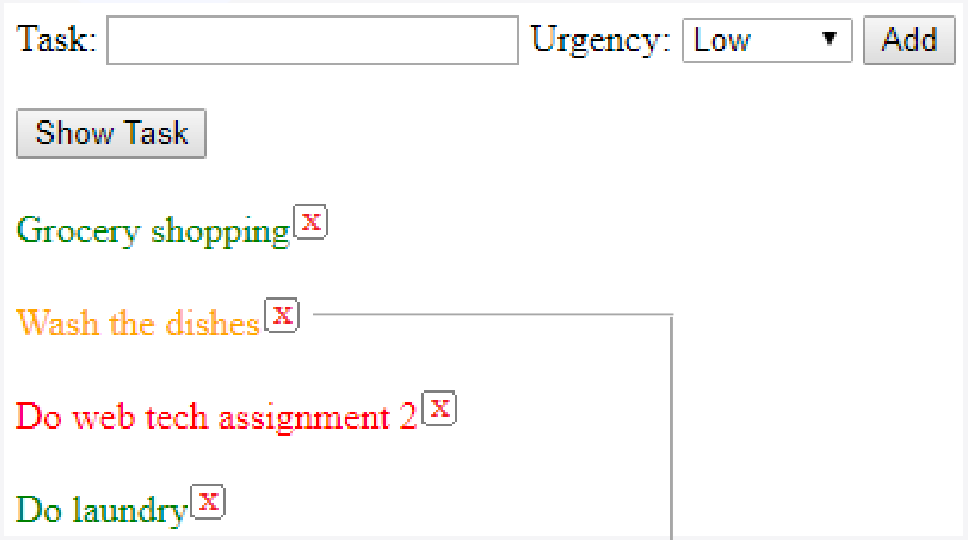
Wash the dishes

Do web tech assignment 2

Do laundry

Example: To-Do-List (3)

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



Grocery shopping

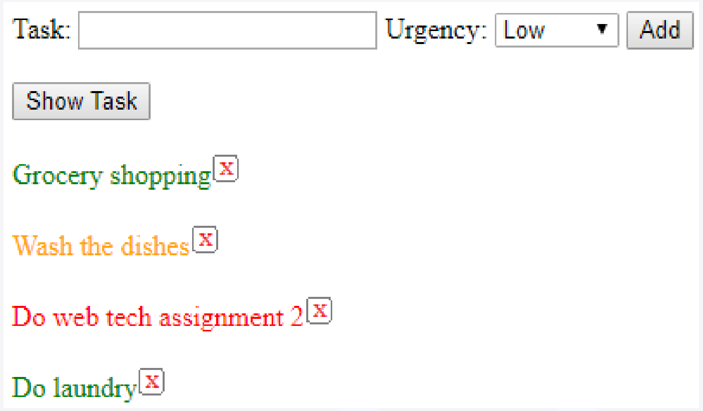
Wash the dishes

Do web tech assignment 2

Do laundry

```
<img src='delete.png' onClick='deleteTask(1605573037767)'/>
```

Example: To-Do-List (3)


Task: Urgency: Low Add

Show Task

Grocery shopping

Wash the dishes

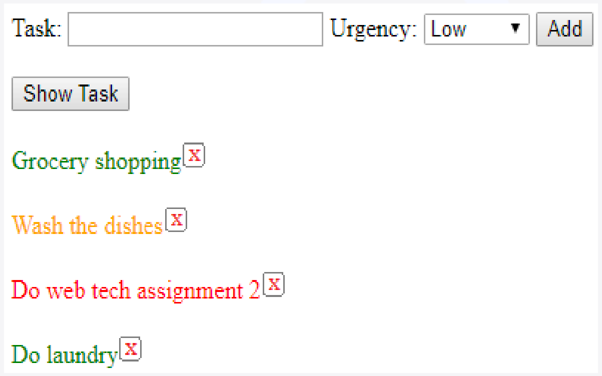
Do web tech assignment 2

Do laundry

```
// 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.time + ")'/>";
     html += "<br /><br />";
  }

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

Example: To-Do-List (3)

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

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

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

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

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


Task: Urgency: Low Add

Show Task

Grocery shopping x

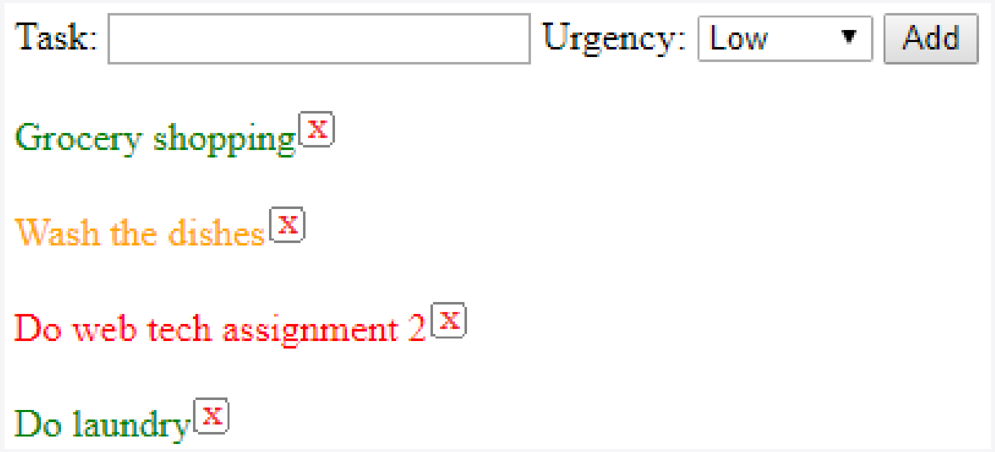
Wash the dishes x

Do web tech assignment 2 x

Do laundry x

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.



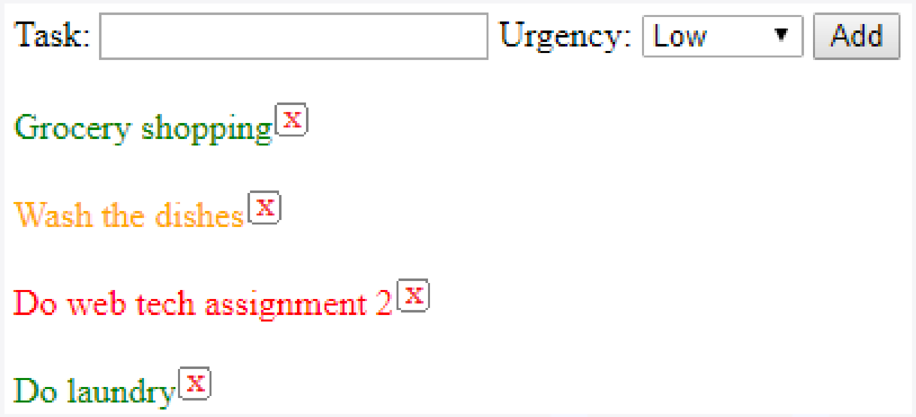
Grocery shopping X

Wash the dishes X

Do web tech assignment 2 X

Do laundry X

Example: To-Do-List (4)


Task: [ ] Urgency: Low ▾ Add

Grocery shopping [x]

Wash the dishes [x]

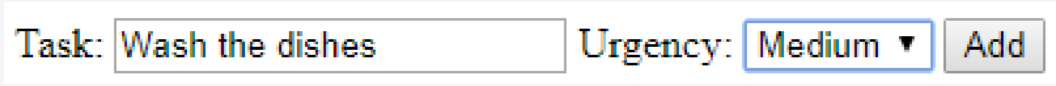
Do web tech assignment 2 [x]

Do laundry [x]

```
<body onLoad="initApp();">
…
// initialize the application
function initApp(){
  if (storageSupported()){

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

    // parse the JSON to the toDoList

    // 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:



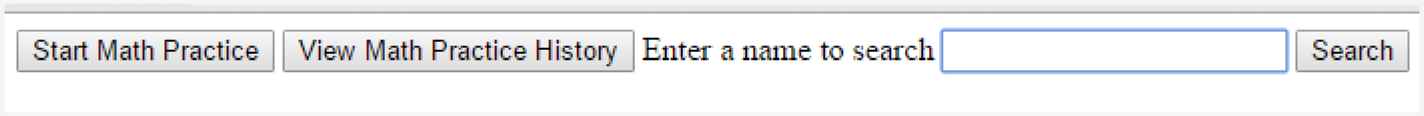
```
// add a task
function addTask(){
  // get task description from user input
  ...

  // add the task to toDoList
  ...


  // if Web Storage supported then update the JSON
  if (storageSupported()){
    ...
  }

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

Example: To-Do-List (4)

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

   // search for the deleted task through the list
   ...

   // if Web Storage supported then update the JSON
   if (storageSupported()){
      ...
   }

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

Example: Math Practice

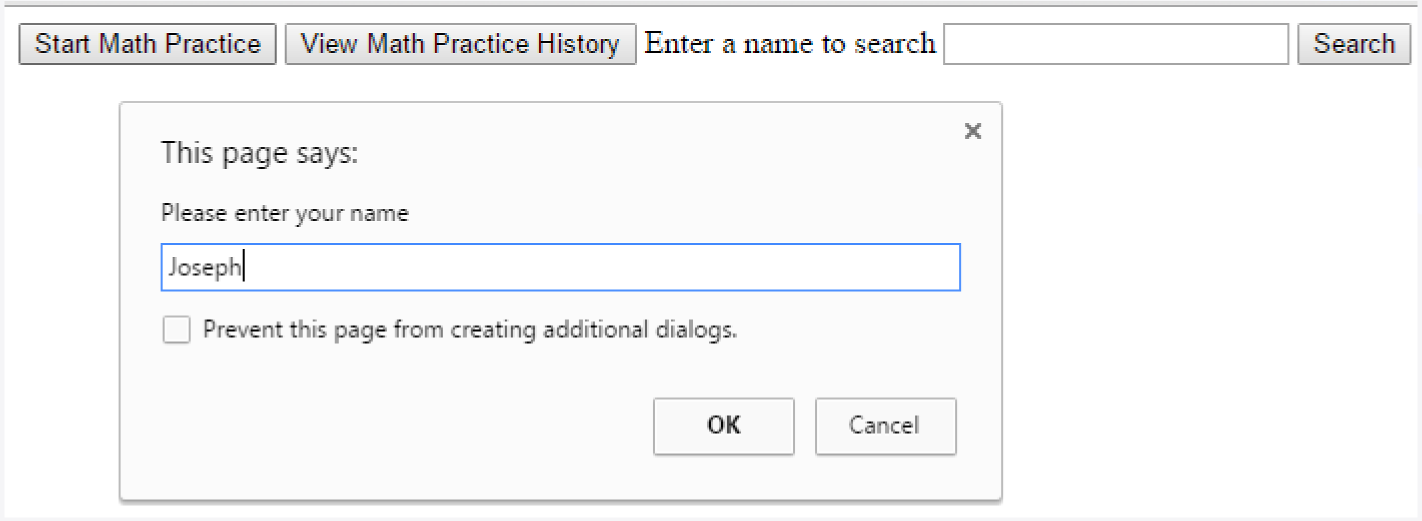
We want to create a web application where children can practice mathematics and parents can view the result of their kids practice.



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.



This page says:

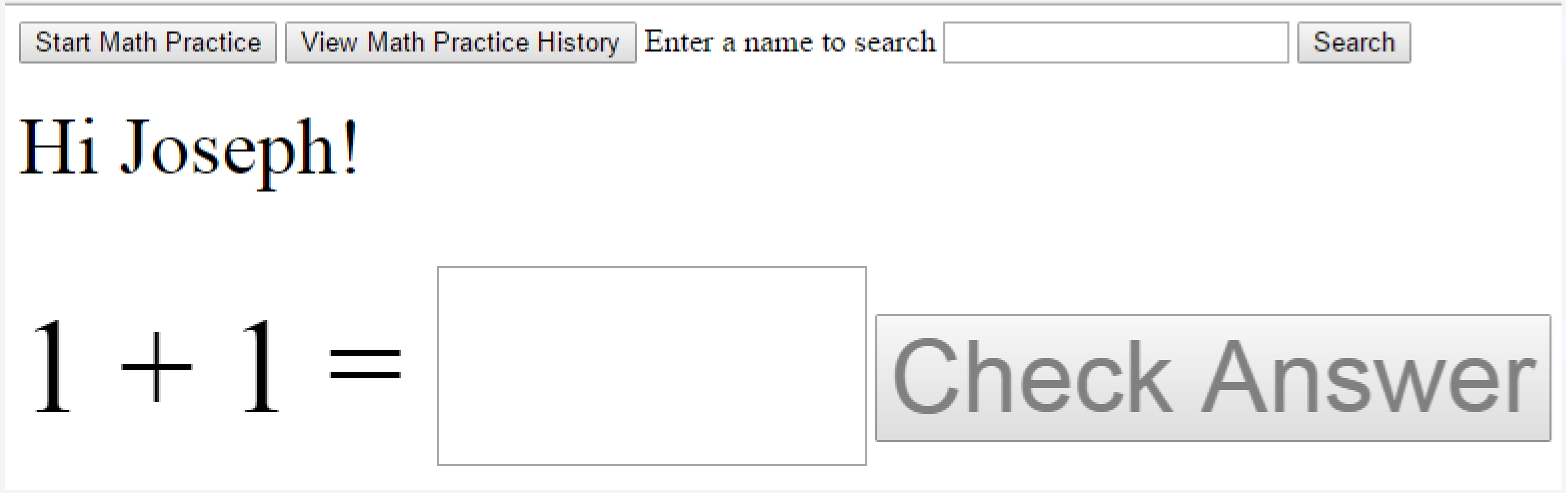
Please enter your name

Joseph

Prevent this page from creating additional dialogs.

Example: Math Practice

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



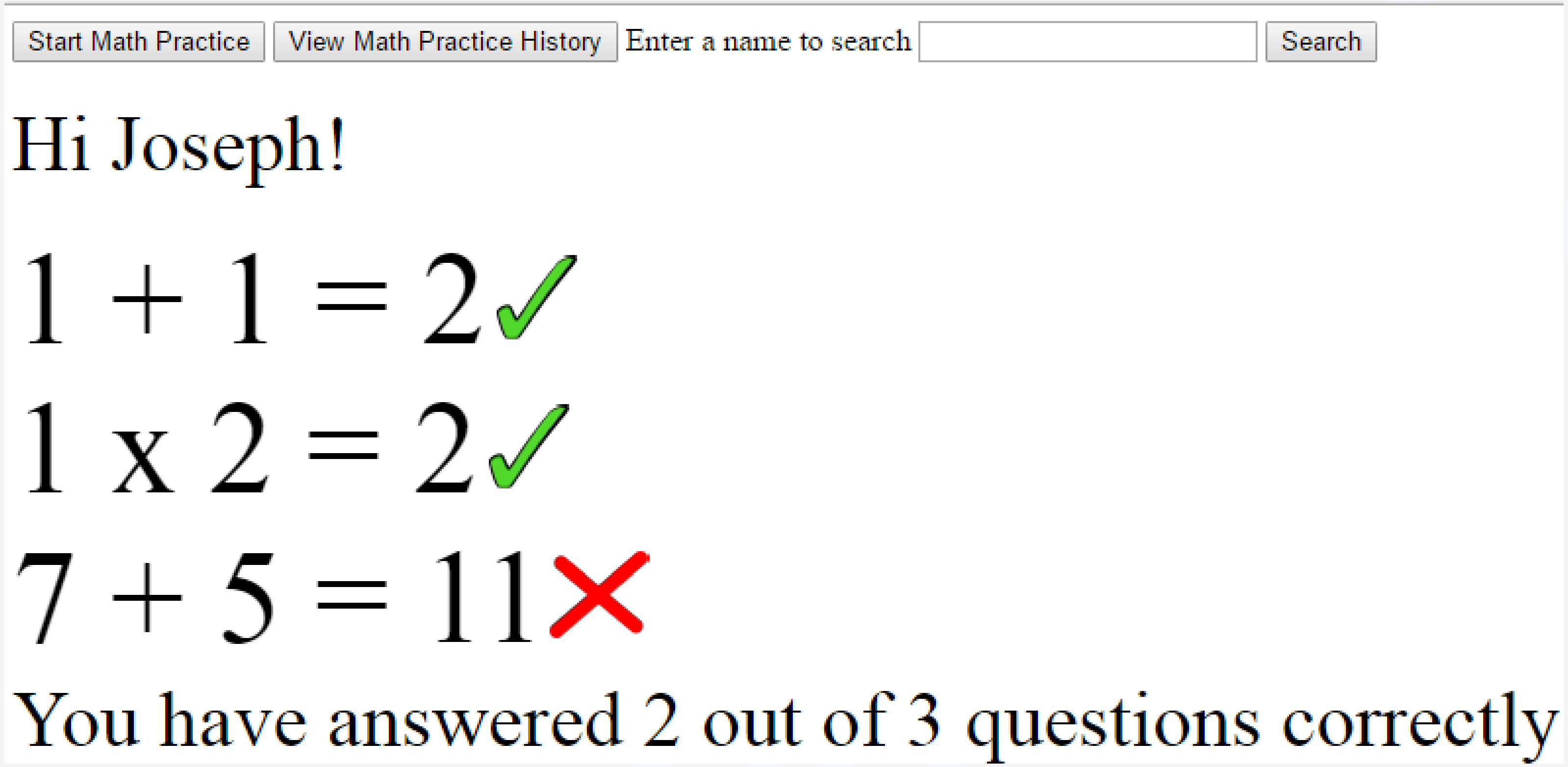
Hi Joseph!

1 + 1 =

Example: Math Practice

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

New question will be generated.



Hi Joseph!

1 + 1 = 2 ✓

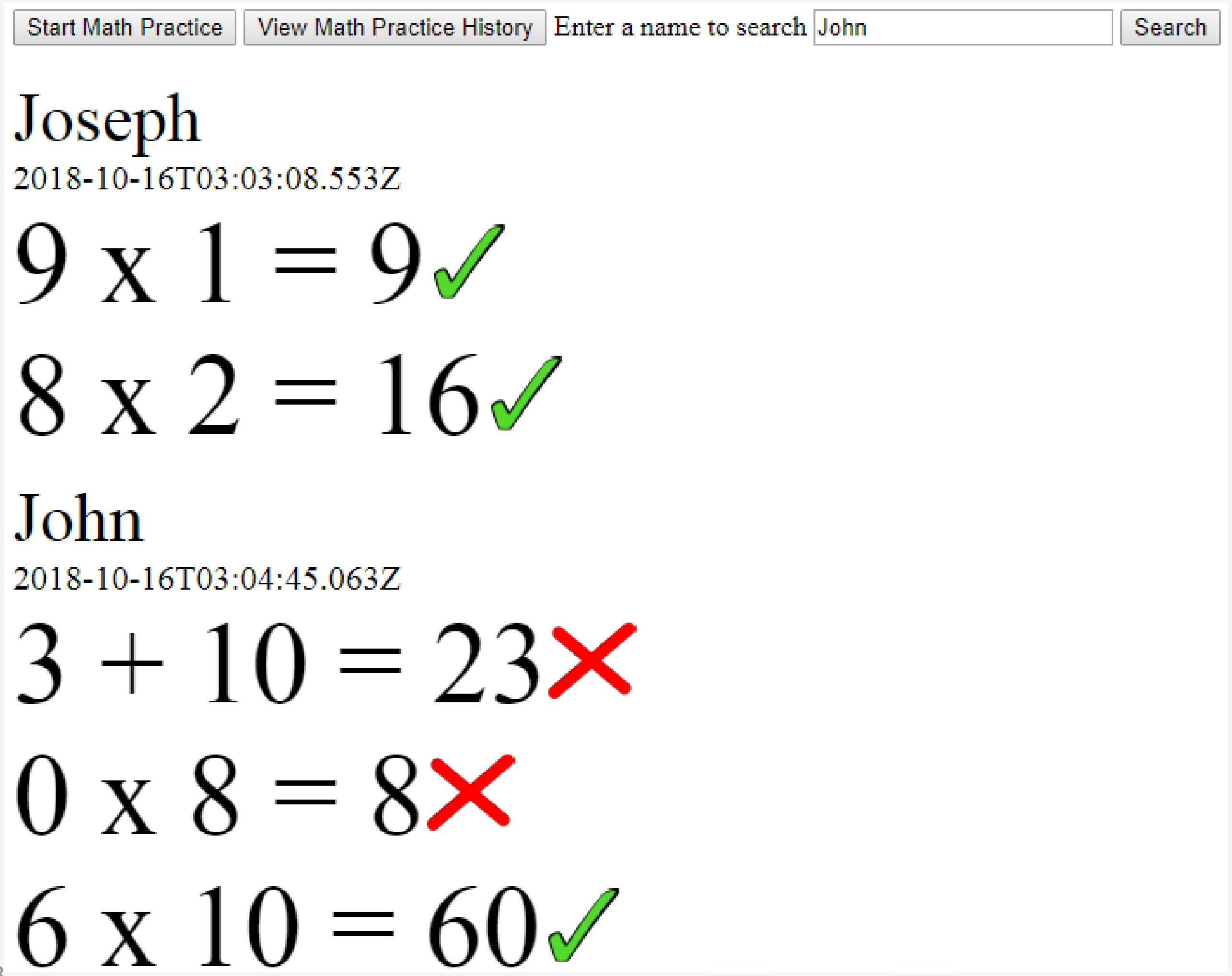
1 x 2 = 2 ✓

7 + 5 = 11 ✗

You have answered 2 out of 3 questions correctly

Example: Math Practice

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



Joseph

2018-10-16T03:03:08.553Z

9 x 1 = 9✓

8 x 2 = 16✓

John

2018-10-16T03:04:45.063Z

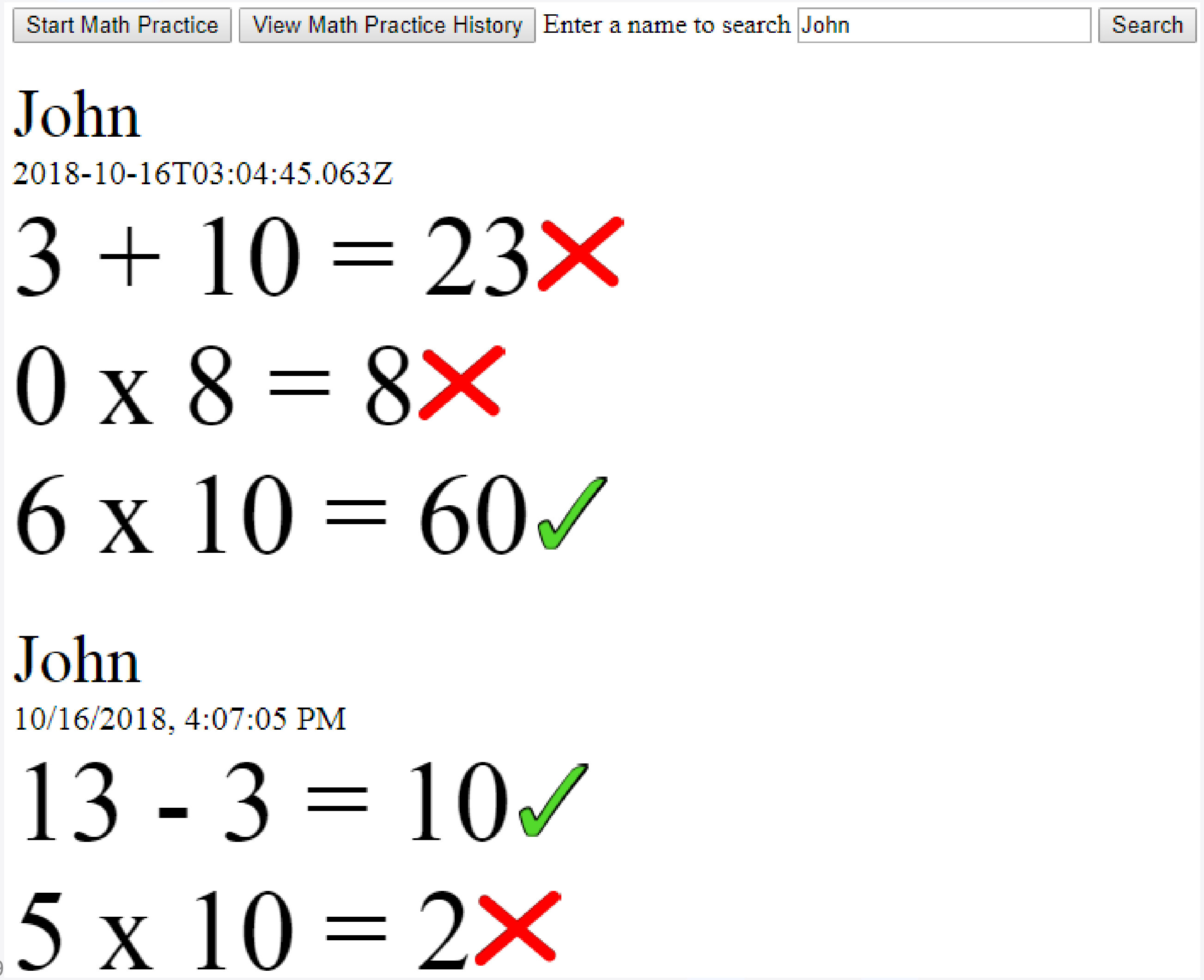
3 + 10 = 23✗

0 x 8 = 8✗

6 x 10 = 60✓

Example: Math Practice

Parents can enter a name to search



Enter a name to search John

John

2018-10-16T03:04:45.063Z

3 + 10 = 23✗

0 x 8 = 8✗

6 x 10 = 60✓

John

10/16/2018, 4:07:05 PM

13 - 3 = 10✓

5 x 10 = 2✗