Lab Guide for Javascript

Storage, location and JSON

The following tutorial has most of the topics for this guide: https://www.w3schools.com/js

You can use the tutorial as a reference guide but can also read it before starting if you prefer a better grasp of the fundamentals before putting them into practice.

The objective of this guide is to add some functionality to the web pages of the previous tutorials using some of the concepts of javascript already covered in previous guides and some new concepts:

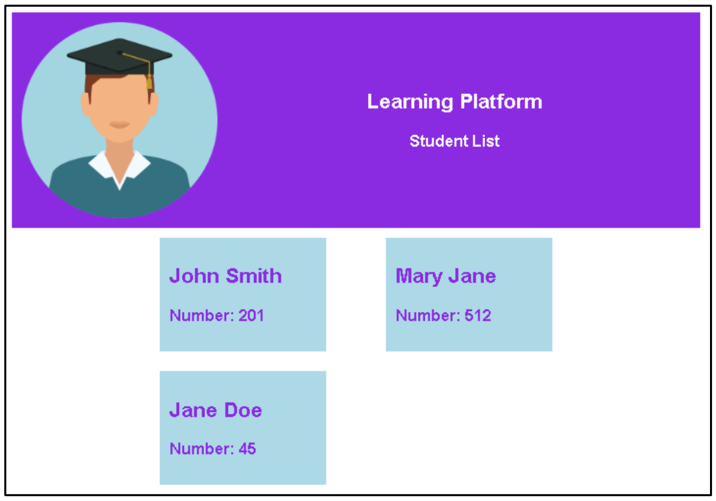
- Web Storage: https://www.w3schools.com/html/html5_webstorage.asp
- Browser Object Model – Window Location: https://www.w3schools.com/js/js_window_location.asp
- JSON: https://www.w3schools.com/js/js_json_syntax.asp
  - Data types: https://www.w3schools.com/js/js_json_datatypes.asp
  - Parse: https://www.w3schools.com/js/js_json_parse.asp
  - Stringify: https://www.w3schools.com/js/js_json_stringify.asp
- Form Validation: https://www.w3schools.com/js/js_validation_api.asp

Open the project that you used for the previous lab guides ("alunos" project unless you choose another name). Notice you should finish previous guides before making this one or at least the previous Javascript tutorial.

1. Create a new page that will show the list of students (students.html, students.css, students.js) :

- In the javascript create a simple list with 3 students with the following information:
  - John Smith with number 201 and id 12
  - Mary Jane with number 512 and id 31
  - Jane Dow with number 45 and id 3
- Use the HTML code bellow and create the javascript and CSS to create result also shown bellow:
  - You can obtain a different appearance, as long as you show the students in their rectangles

```
<html>
  <head>
  <title>Learning Platform</title>
  <link rel="stylesheet" href="stylesheets/style.css">
  <link rel="stylesheet" href="stylesheets/students.css">
  <script src="javascripts/students.js"></script>
  </head>
  <body>
      <header>
      <a href="index.html">
       <img id="logo"
         src="images/student.png"
         alt="student image"></a>
      <section>
          <h1>Learning Platform</h1>
          <h3> Student List  </h3>
      </section>
      </header>
      <main id="students">
      </main>
  </body>
</html>
```


Learning Platform

Student List

John Smith

Number: 201

Mary Jane

Number: 512

Jane Doe

Number: 45

2. Now we will allow the user to click the student rectangles and go to the student grade information

- When you create the student HTML define the onclick property for the container. The onclick value should be a call to the showStudent method with the position of the array where the student information is.
- Create the showStudent method with position parameter. This method should:
  - Save the array position received using “sessionStorage.setItem” (ex: field name studentId)
  - Save the name of the student (same command but different field name, ex: studentName)
  - Change location to the page studentGrades.html, to show the grades of the chosen student.

Remember how to save an option chosen by the user:

- If you have a list of objects:
```
var items= [
    {  product_id: 12, product_name: "milk", total: 3.6 },
    {  product_id: 2, product_name: "eggs" , total: 10.08 },
    {  product_id: 3, product_name: "chicken", total: 0.92}
 ]
```
- You can create the corresponding HTML in the window.onload function and include for each element a onclick that will run code specific to the object using the position of the array:
```
window.onload = function() {
   let html ="";
   for (let i in items)
     html += "<section onclick='showItem("+i+")'>"+
        "<h3>"+items[i].name+"</h3>"+
        "<p> Price "++items[i].total+" €</p>";
   document.getElementById("items").innerHTML = html;
 }
```
- You then need to create the showItem function, that will be called when the user clicks on the section, and will receive the position of the corresponding item
```
function showItem(pos) {
    sessionStorage.setItem("itemPos",pos);
    sessionStorage.setItem("productName",items[pos].name);
    window.location = "item.html";
 }
```

3. We now need to change the studentsGrading.js (.html and .css will be the same) so that the information about the chosen student is shown (insted of a fixed student)

- Change the data structure. The data structures will now have a list of lists of units.
  1. The units will follow the same structure
  2. Each list corresponds to the list of units for a student
  3. The order is the same as the previous list of students i.e. the first list of units corresponds to the list of units of the student in the first position of the previous student array

  Keep the existing list for the first student and for the other students create 2 lists with 3 and 2 units respectively.

(continues in the next page)

- Change the onload function to read the name and position of the student and show the information of that student. You only need:
    i. To use the read student name instead of the variable that was there before
    ii. To obtain the array of units that corresponds to the chosen student (the array at the position read from the web storage)
    iii. To use that array of units instead of the array that was used before

Remember Arrays and reading from Web Storage:

- An array can have any type of values inside, including other arrays:

```
var baskets = [ [“Potatoes”,”Onion”], [“Pork”,”Eggs”,”Butter”], [“Apples”]];
```

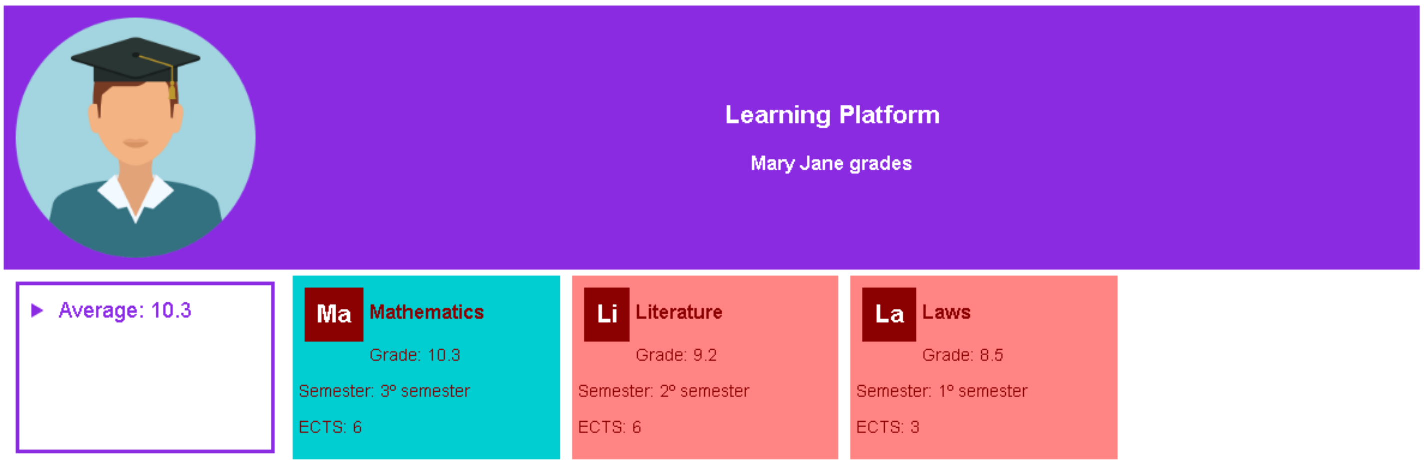
- You can retrieve values from web storage using the getItem method:

```
window.onload = function() {
  let pos = sessionStorage.getItem("itemPos");
  let name = sessionStorage.getItem("productName");
  …
}
```

- You can retrieve the list you want to an auxiliary variable and then access it as a “normal” array

```
let basket = baskets[pos];
let html = “”;
for (let item on basket)
…
```

Expected result for the unit grades list when the second student is chosen (the values will vary since they depend on what values you placed inside the list for that student, but the number of units shown should be 3, the name should also be the same).



4. Now we will change the Grading functionality. First include 3 new paragraphs:

- One after the line where the Project grade and percentage is. It will be used to show an error message if we have an invalid value for the project grade or percentage
- One after the line where the Test grade and percentage is to do the same for the test grade and percentage.
- One after the previous paragraph to show an error when the sum of percentages is different then 1.0

All these paragraphs are initially not seen (display:none) and when shown it will have red letter color

Tip: use CSS classes to apply this styling (you need to create a CSS file for this styling since there probably is not one specific to this file)

5. Change the grading.js to make the verifications and show the error when they fail

- You should create a function for each verification that will return true or false when the verification passes or fails (respectively). You may want to receive an object and save the value verified on that object also.
- The verifications will be called in the function associated with the button (called on the onclick)
- The function of the button will exit if any verification fails. If all verifications pass, we should have an object will all values for the next step

Remember verifying values of form elements:

- The button that submits the form will call a method that will make all verifications and submit:
```
<input type="button" value="Submit" onclick="verifyAndSubmit()">
```

- In the HTML you can also have elements that will show the errors. Usually these elements are initially hidden by using display:none in their CSS, and will be shown using javascript only when an error occurs

- In the method you can call each verification and create an object with all values:

Different from the slides

```
function verifyAndSubmit() {
   let obj = {};
   let correct = true;
   if (!verifyAndSaveName(obj)) correct = false;
   if (!verifyAndSavePrice(obj)) correct = false;
   if (!correct) return;  // exiting only after all verifications have run
   …                      // this way all errors will be shown
```

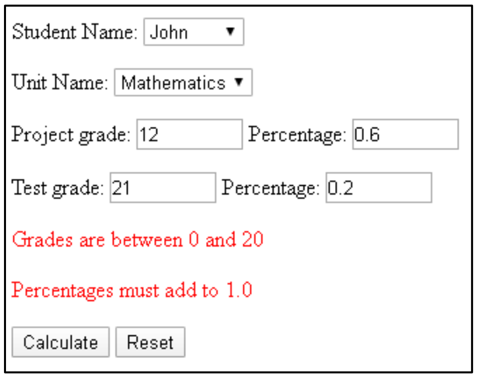
- The function that verifies will fill the errors and save the values, returning true in case of success and false otherwise
```
function verifyAndSaveName(obj) {
   obj.name =  document.getElementById("name").value;
   let ename = document.getElementById("errorName");
   if (obj.name.length < 3) {
      ename.innerHTML = “Name is too short”;
      ename.style.display = “block”;
      return false;
   }
   ename.style.display = “none”; // hiding the error if there is no error anymore
   return true;
}
```

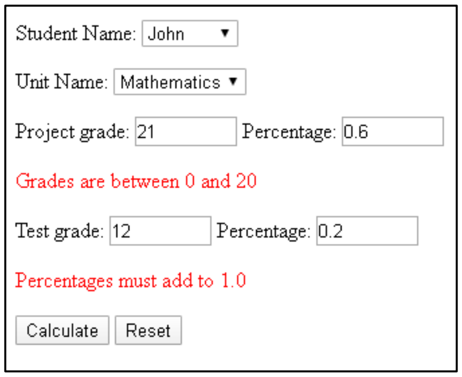
Remember form verifications:

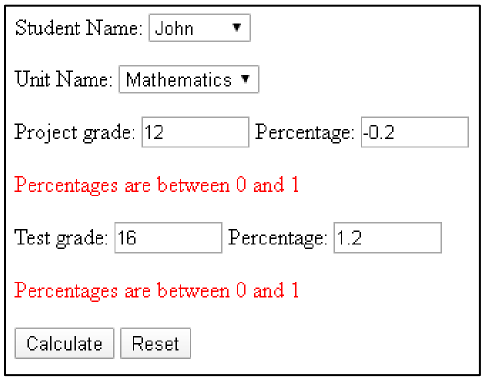
- For some inputs, like numeric inputs, you can automatically use the embedded verification (ex: if you set the max and min properties)
  We will need to use the “checkVality” method of the form elements object

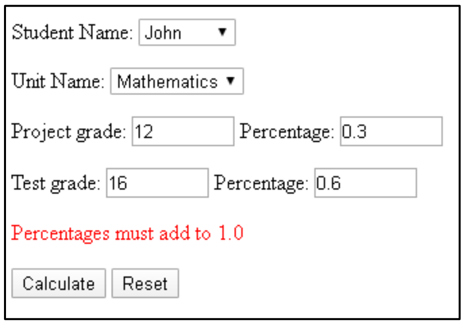
```
function verifyAndSavePrice(obj) {
   let price = document.getElementById("price");
   let eprice = document.getElementById("errorPrice");
   if (!price.checkValidity()) {
      eprice.innerHTML = “Please use a price value greater than 0”;
      eprice.style.display = “block”;
      return false;
   }
   obj.price = price.value; //Can also save before the if, it will save invalid values also
   eprice.style.display = “none”; // hiding the error if there is no error anymore
   return true;
}
```

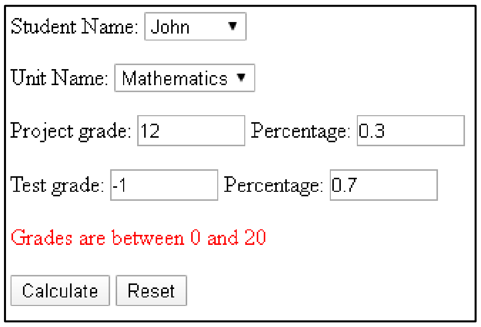
Here are some examples of failed verifications (and last one with all correct values):

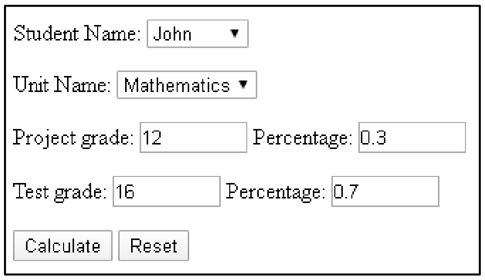

Student Name: John
Unit Name: Mathematics
Project grade: 12 Percentage: 0.6
Test grade: 21 Percentage: 0.2
Grades are between 0 and 20
Percentages must add to 1.0


Student Name: John
Unit Name: Mathematics
Project grade: 21 Percentage: 0.6
Grades are between 0 and 20
Test grade: 12 Percentage: 0.2
Percentages must add to 1.0


Student Name: John
Unit Name: Mathematics
Project grade: 12 Percentage: -0.2
Percentages are between 0 and 1
Test grade: 16 Percentage: 1.2
Percentages are between 0 and 1


Student Name: John
Unit Name: Mathematics
Project grade: 12 Percentage: 0.3
Test grade: 16 Percentage: 0.6
Percentages must add to 1.0


Student Name: John
Unit Name: Mathematics
Project grade: 12 Percentage: 0.3
Test grade: -1 Percentage: 0.7
Grades are between 0 and 20


Student Name: John
Unit Name: Mathematics
Project grade: 12 Percentage: 0.3
Test grade: 16 Percentage: 0.7

6. Complete the function by saving the object in the session and calling a new page that will show the filled details
   - Convert the object to JSON using JSON.stringify
   - Save the JSON using sessionStorage.setItem
   - Change the window.location to the new page

7. Create a new page that will show the information that you created on the previous page:
   - Create an HTML with elements that will be filled with the information you need
     i. This HTML will be a template that will be filled with the information in the form
     ii. You should not create the full HTML on the onload function, only insert the information that is variable, all fixed content will be in the HTML (thus being a template)
   - The new page does not need to have any CSS (only if you want to)
   - Read the information saved in the previous page using sessionStorage.getItem
   - Parse the JSON to a Javascript object using JSON.parse
   - Use the object property values to fill the missing spaces in the HTML template

Remember writing and reading from Web Storage:

- For saving and object you first need to convert it to JSON and then use setItem:

```
function verifyAndSubmit() {
   let obj = {};
   …
   let json = JSON.stringify(obj); // converting object to json
   sessionStorage.setItem(“item”,json); // saving json on Web Storage
   window.location = “item.html”; // changing  to the item page
}
```

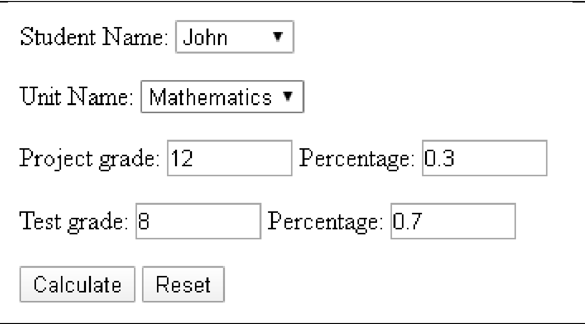
- For reading the object in the next page you now use getItem and then convert the value from JSON back to a Javascript object:

```
window.onload = function () {
   let json = sessionStorage.getItem(“item”);
   let item = JSON.parse(json);
   …
```

Final example of result in the next page

Example of the result

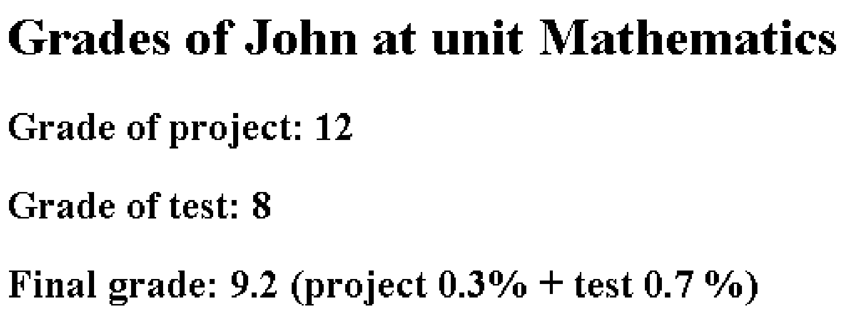
Example of form and the created JSON:



Student Name:	John ▾		
Unit Name:	Mathematics ▾		
Project grade:	12	Percentage:	0.3
Test grade:	8	Percentage:	0.7

```
‘{
    "proj":{"grade":12,"percentage":0.3},
    "test":{"grade":8,"percentage":0.7},
    "student":{"name":"John","number":201},
     "unit":{"name":"Mathematics","semester":3,"ects":6}
 }’
```

Example of the final page:



Grades of John at unit Mathematics

Grade of project: 12

Grade of test: 8

Final grade: 9.2 (project 0.3% + test 0.7 %)

8. If you want, you can now improve this final page and the previous one with CSS.
You can also include an extra line saying if the student has passed the unit or not

The End