

Lab Guide for Javascript

Arrays, objects, loops and functions

The following tutorial has most of the topics for this guide and the next guides on javascript: <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:

- Objects in Javascript: https://www.w3schools.com/js/js_objects.asp
- Arrays in Javascript: https://www.w3schools.com/js/js_arrays.asp
 - Methods: https://www.w3schools.com/js/js_array_methods.asp
- Loops:
 - The for loop: https://www.w3schools.com/js/js_loop_for.asp
 - Break and continue: https://www.w3schools.com/js/js_break.asp
 - The while and do_while loops: https://www.w3schools.com/js/js_loop_while.asp
- The select element (HTML): https://www.w3schools.com/tags/tag_select.asp
- The details element (HTML): https://www.w3schools.com/tags/tag_details.asp

NOTE: The topics were presented in the order they will be approached in the guide which is not always the same used in the w3schools tutorial.

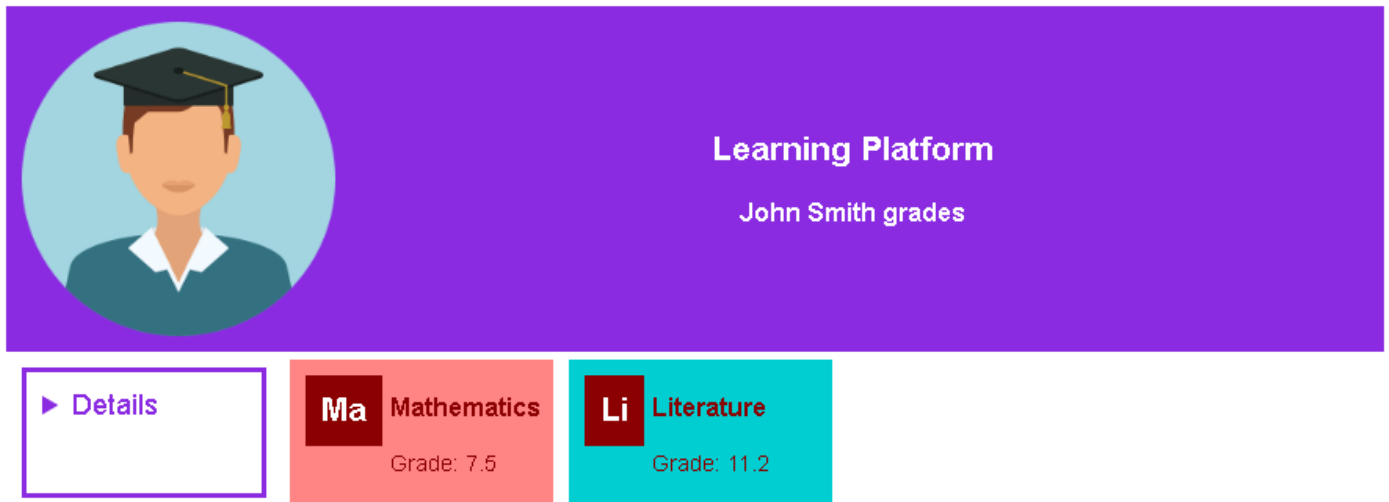
Open the project that you used for the previous lab guides ("alunos" project unless you choose another name). Notice you should finish previous guides before making this one or at least the previous Javascript tutorial.

1. Open the studentGrades files (.html, .css and .js) and do the following modifications:
 - In the studentGrades.html create a details element and give it an id. For now, it will be empty, but we will use it for placing the current average score (in the summary) and the number of units finished and failed.
 - Correct the grid layout to include the details element
 - We gave details the nav area in the grid
 - We will edit the javascript in the next steps

```
...  
</header>  
<details id="summary"></details>  
<main id="grades">  
...
```

NOTE: The details element will automatically place a message "Details" when empty.
When we place a summary element the summary text will show instead of the word Details
The rest of the contents only show if we click the summary text

Expected layout result:



2. In the javascript file we will keep the student name variable but will delete all other variables, changing them for a single variable with an object list. Each object should represent the information of a unit with the name, the grade for that student, the semester and the number of ECTS. Create an object array with the following information:
- Unit Mathematics, grade 8.3, 3^o semester, 6 ECTS
 - Unit Literature, grade 11.2, 2^o semester, 6 ECTS
 - Unit Laws, grade 18.5, 1^o semester, 3 ECTS
 - Unit Informatics, grade 14.3, 1^o semester, 6 ECTS
 - Unit Cooking, grade 7.4, 2^o semester, 3 ECTS

Remember Objects:

- An object represents an entity and holds the information about that entity instance. It is declared by placing pairs of property_name : property_value inside { }. An example of an object that represents a client:

```
var johnInfo = {
  name: 'John Smith',
  yearBorn: 1980,
  height: 1.79,
  vip: true
};
```
- To access object information, you must access the place where it was storage and then use its properties;

```
let name = johnInfo.name;
let isVip = johnInfo.vip;
```

(More information on the next pages)

Remember Arrays:

- An array is used to store multiple values. To declare it just place the values inside [] separated by commas:

```
var names = ["John", 'Jade', "Jill", "Joana"];
```

 - Javascript arrays can have different types of values in the same array (not common to happen)

```
var stuff = ["Text", 12, 12.4, true, new Date(), {x:12,y:-2} ];
```
- You can access the information in the array by using its index. Array index start at zero, and you can know the size using the `length` property. The last element as index `length-1` since the index start at zero.

```
let firstName = names[0];  
let lastName = names[names.length-1];
```
- You can also use `pop` and `push` to add or remove elements from the end (or `shift` and `unshift` to do the same from the start):

```
let lastName = names.pop();  
names.push("Joseph"); // names now ["John", 'Jade', "Jill", "Joseph"];
```

Remember Iterating over an array:

- You can use a for loop to go through the array:

```
let namesStr = "";  
for(let i=0; i<names.length; i++) {  
    namesStrt += "<p>"+(i-1)+"-"+names[i]+"</p>";  
}
```

Result would be all the names preceded by an order number inside paragraph elements.

You can do the same using a `for in`:

```
let namesStr = "";  
for(let i in names) {  
    namesStrt += "<p>"+(i-1)+"-"+names[i]+"</p>";  
}
```

Or if you do not need the index you can use the `for of`:

```
let namesStr = "";  
for(let name of names) {  
    namesStrt += "<p>"+name+"</p>";  
}
```

(More information on the next page)

Remember Objects with arrays and arrays of objects:

- You can use arrays inside objects

```
var student = { number: 20191234; grades: [12, 8.5, 15, 16]; }
```

- Or objects inside arrays. To obtain a value of the object you need first to obtain the correct object in the array using the index and then the correct object property:

```
var productCart = [  
  { name: "bananas", quantity: 0.5, unitPrice: 1.03 },  
  { name: "6 eggs", quantity: 2, unitPrice: 0.93 },  
  { name: "1l Milk", quantity: 6, unitPrice: 0.54 } ];  
let totalPrice = 0;  
for (let product of productCart) { totalPrice += product.unitPrice; }
```


Or

```
for (let i=0; i < productCart.length; i++) {  
  let product = productCart[i];  
  totalPrice += product.unitPrice;  
}
```

3. Now we can use the array of objects to create the HTML instead of the variables. Follow the steps below to create the HTML content of the main.

- Create a function that receives a unit object and returns a string with the HTML of one unit (similar to the HTML we created before for the units), the HTML should also include the information about the semester and number of ECTS. It should also use a class to mark the element as failed (that will be used in the CSS to show a different background color).
- In the window.onload replace the code that generates the string for a loop that calls the function for each unit and joins (concatenates) all the strings into one string. That string will now be used for the innerHTML of the main.
- Notice that this code is now generic, you can add more units to the list and they will be presented in the HTML.

Expected result for the unit grades list (more information on the next page):



Learning Platform
John Smith grades

Details

Ma Mathematics

Grade: 8.3

Semester: 3rd semester

ECTS: 6

Li Literature

Grade: 11.2

Semester: 2nd semester

ECTS: 6

La Laws

Grade: 12.5

Semester: 1st semester

ECTS: 3

In Informatics

Grade: 14.3

Semester: 1st semester

ECTS: 6

Co Cooking

Grade: 7.4

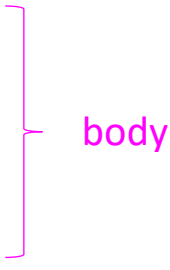
Semester: 2nd semester

ECTS: 3

Remember Functions:

- A function is declared using the **keyword function** followed by the **name**, **the list of parameters** and the **body** of the function. The list of parameters is a list of names separated by commas. Example:

```
function createProductHtml( product ) {  
  let html = "<section>";  
  let total = product.quantity * product.unitPrice;  
  html += "<h1>"+product.name+"</h1>";  
  html += "<p> Total cost: "+ total + "</p>";  
  html += "</section>";  
  return html;  
}
```



In this case the function has only one parameter, the product object, and returns the HTML string created from the product data.

- A function can be used in any object, or called for an array of objects, for instance for the **productCart** array:

```
window.onload = function () {  
  let html = "";  
  for (let product of productCart) {  
    html += createProductHtml(product);  
  }  
  document.getElementById("elemId").  
    innerHTML = html;  
}
```

We can also use this for:

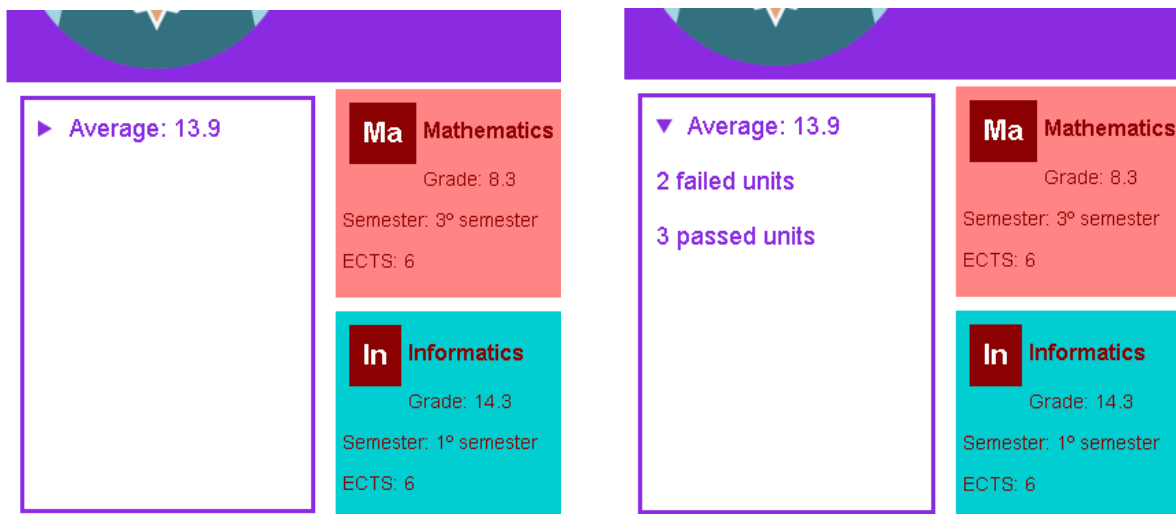
```
for (var i=0; i< productCart.length; i++) {  
  let product = productCart[i];  
  html += createProductHtml(product);  
}
```

The product variable will hold one product of the array in every loop cycle and we will call createProductHtml for each of them. The return value of each call will be concatenated into a single string that will be placed inside the innerHTML of the element with id "elemId" and be shown in the HTML.

NOTE: The function is declared outside the onload because all the values it needs are received as a parameter (it would also be the same if the values were global variables), but the for needed to be placed inside the onload because after it we will need to access the DOM and we need to make sure the DOM tree was already loaded.

4. Now to calculate the information to place on the details element.
- In the window.onload block create three variables:
 - sum of grades that student finished
 - you can weight them by number of ECTS and in that case you need to have the sum of ECTS
 - number of units failed
 - number of units the student finished.
 - Inside the loop you will need to update these variables.
 - After the loop you will use these variables to create:
 - A summary element with the average grade, calculated using the sum of grades and number of units finished (if you weighted the grades by ECTS you use the sum of ECTS instead of the number of units)
 - A paragraph with the number of units finished
 - A paragraph with the number of units failed

This is the expected result for the details, before and after clicking the summary:



The average was calculated using the ECTS values as weights.

We also limited the number of digits after the floating point to 1 (see the javascript function toFixed()).

The average without using weights is 14.7

5. Now we will update the grading.js file.

- Create two lists:
 - List of student objects each one with a name and a number
 - List of unit objects each one with a name, a semester and the number of ECTS.
- Replace the input for the student and for the unit for selects.
- Fill the select with options from the lists you created, you can use only the name of each object or create a function to transform the object into the HTML string like before, the format of the string is up to you.
- The rest of the code can stay the same since the “value” property also exists on the select element

Example of the result using only the name for units and students (more information in the next page also):

Student Name: Mary
Unit Name: Literature
Project grade: 12 Percentage: 0.4

Unit Name: Literature
Project grade: 12 Percentage: 0.4

Student Name: Mary
Unit Name: Literature
Project grade: 12 Percentage: 0.4
Test grade: 9 Percentage: 0.6
Calculate Reset
Student Mary obtained 10.2 on the Literature unit.

Remember Fill a select with values from an array of objects:

The select element will show a combobox where the user can choose from several options.

It is declared using the <select> tag with <option> tags inside for each option. Inside the <option> tag we will place the text that will be shown.

We can also use the “value” property to define a value for the option, that way the user will see the text but the value obtained from the select will be the one on the value property, for instance the index of the element on the list. If no “value” property is defined the value is the same as the text inside the option.

If you have a list of objects you can create a select like this:

```
let listClients = [  
  {name: "John", vip: false },  
  {name: "Mary", vip: true },  
  {name: "Anthony", vip: false }  
];  
  
window.onload = function () {  
  let html = "";  
  for (let client of listClients) {  
    html+=""+client.name+"</option>";  
  }  
  document.getElementById("elemId").innerHTML = html;  
}
```

Remember Using the index in the select to obtain the full object from the selected option:

If you need the select to return a different value than the one shown, the index for instance, just change the for to something like this:

```
for (let i in listClients) {  
    html+="}
```

In this case we used the **for in** since we needed the index to use for the value, the text is still the name, but now we needed to obtain the object from the array using the index before retrieving the name property value.

In this case when you retrieve the value from the select element you will get the index and must use that to obtain the values from the array:

```
let index = document.getElementById("elemId").value;  
let client = listClients[index];  
let html = "";  
if (client.vip) {  
    html += "VIP ";  
}  
html += "Client " + client.name  
document.getElementById("result").innerHTML = html;
```

Result if John is chosen:

Client John

Result if Mary is chosen:

VIP Client Mary

6. You can change the previous code so that instead of showing the name of the selected student and unit the final text shows all the information of the unit and student selected. You will need to:

- Use the "value" property in each <option> with the position of the element on the list
- After obtaining the index value chosen you can now obtain the full object from the list
- Create a string from the object that shows all the information.

Example of the result:

Student Name:

Unit Name:

Project grade: Percentage:

Test grade: Percentage:

Student Mary with number 202 obtained 11.0 on the Laws unit (3 ETCS) of the 1º Semester

The End