Lab Guide for Javascript

DOM, variables, conditionals and forms

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:

- Script Tag: https://www.w3schools.com/js/js_whereto.asp
- DOM: https://www.w3schools.com/js/js_htmldom.asp
  - innerHTML: https://www.w3schools.com/js/js_htmldom_methods.asp
  - Document object: https://www.w3schools.com/js/js_htmldom_document.asp
  - Changing CSS: https://www.w3schools.com/js/js_htmldom_css.asp
- Variables: https://www.w3schools.com/js/js_variables.asp
  - Let: https://www.w3schools.com/js/js_let.asp
  - Const: https://www.w3schools.com/js/js_const.asp
- Operations (numeric): https://www.w3schools.com/js/js_operators.asp
  - Arithmetic: https://www.w3schools.com/js/js_arithmetic.asp
  - Assignment: https://www.w3schools.com/js/js_assignment.asp
- Data types in javascript: https://www.w3schools.com/js/js_datatypes.asp
- Strings: https://www.w3schools.com/js/js_strings.asp
  - Methods: https://www.w3schools.com/js/js_string_methods.asp
- More on numbers: https://www.w3schools.com/js/js_string_methods.asp
  - Methods: https://www.w3schools.com/js/js_number_methods.asp
- Booleans: https://www.w3schools.com/js/js_booleans.asp
  - Comparisons: https://www.w3schools.com/js/js_comparisons.asp
- Conditions: https://www.w3schools.com/js/js_if_else.asp
- Form inputs (HTML): https://www.w3schools.com/html/html_form_attributes.asp
  - Obtaining and changing the value of inputs: https://www.w3schools.com/jsref/prop_text_value.asp
  - Javascript input manipulation examples: https://www.w3schools.com/js/js_input_examples.asp
- Events: https://www.w3schools.com/js/js_htmldom_css.asp
  - On click event examples: https://www.w3schools.com/jsref/event_onclick.asp
  - **NOTE**: instead of:

    ```
    <button onclick="myFunction()">Click me</button>
    ```
  - You should use

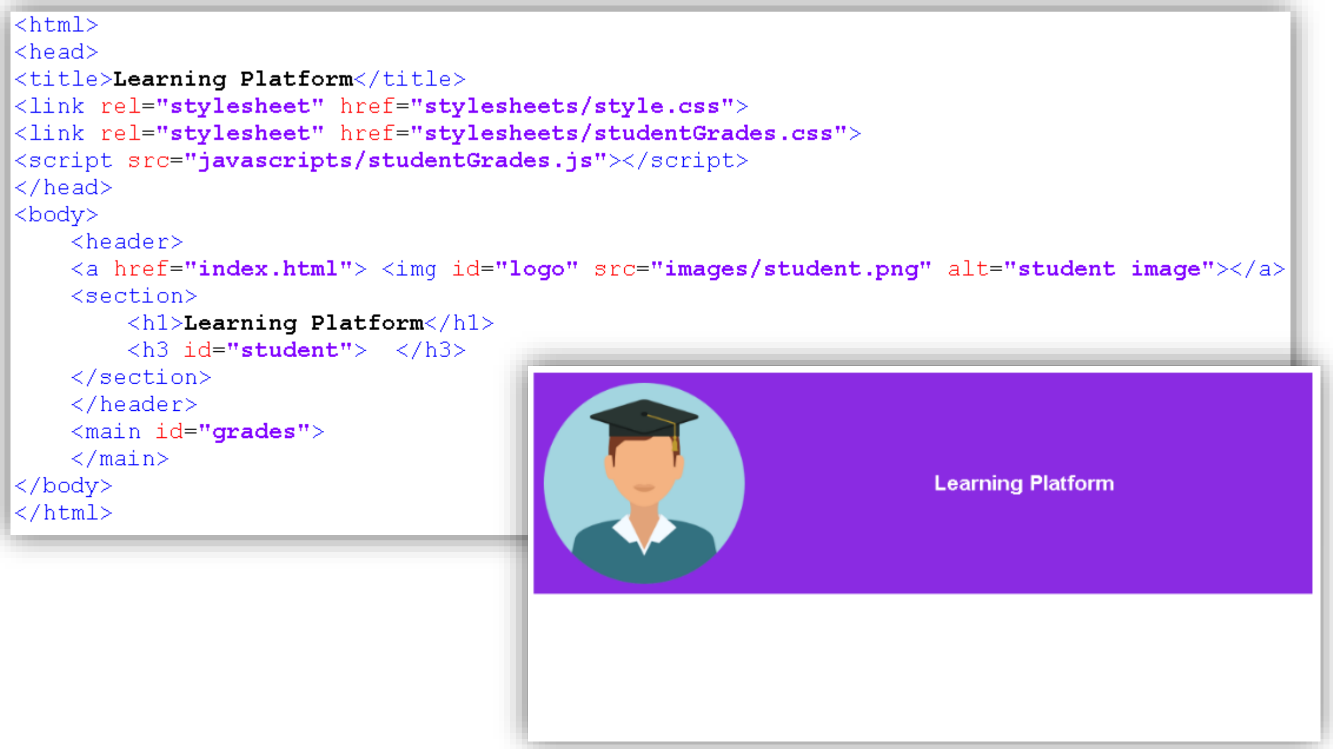
    ```
    <input type="button" value="Click me" onclick="myFunction()">
    ```

NOTE: The topics were presented in the order they will be approached in the guide which is not always the same used in the w3shools tutorial.

Open the project that you used for the HTML and CSS lab guides ("alunos" project unless you choose another name). Notice you should finish both previous guides before making this one, but you can do this tutorial without them since we will mostly work with new pages.

1. In the "public" folder create the following files:

- studentGrades.html will have a header similar to the units.html file and will link to CSS files style.css and studentGrades.css and to the javascript file studentGrades.js. It will also have a main tag where we will create the information about the grades using javascript.
    - The style.css is the one we created before and holds all the general CSS rules that create the default style of our web site
- studentGrades.css will be created on the stylesheets directory and defines all the CSS rules that are specific only to this page
- studentGrades.js will be where we are going to create our javascript code to make calculations and generate the HTML inside the main tag



```
<html>
<head>
<title>Learning Platform</title>
<link rel="stylesheet" href="stylesheets/style.css">
<link rel="stylesheet" href="stylesheets/studentGrades.css">
<script src="javascripts/studentGrades.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 id="student">  </h3>
    </section>
    </header>
    <main id="grades">
    </main>
</body>
</html>
```

We placed two ids:

- student: to place the name of the student
- grades: to add all the grades information

The page is still mostly empty since we want to create this information using javascript.

2. In the javascript file create a variable for the name of the student. Use that variable to fill the subtitle. For instance, if the name is **John Smith**, the subtitle should be **John Smith grades**.

Remember:

- You declare global variables using the var keyword.
  ```
  var number = 12;
  ```

- Text values must be enclosed in “ ”;
  ```
  var text = “Hello”;
  ```

- You can concatenate text using the + operator and assign the result to a variable, or use the += operator:
  ```
  text += “ World”; // text now holds “Hello World”
  ```

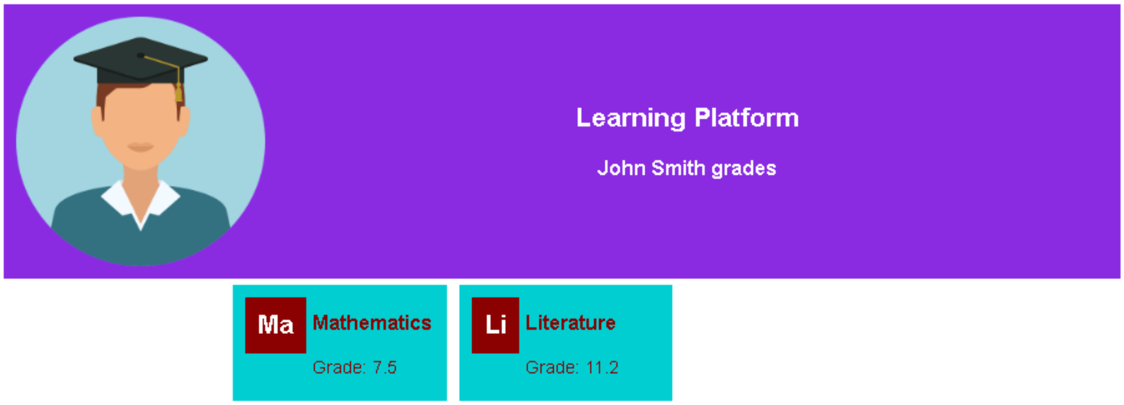
- You can manipulate any part of the HTML using the DOM. You can find elements using their id and change their content:
  ```
  document.getElementById(“theId”).innerHTML = “New text”;
  ```

- When the javascript is included at the beginning the HTML it will run before the HTML was loaded, so we must wait for the HTML to load before we can run commands that manipulate the DOM. We can do this using the window.onload property and setting a function with code that only will be run when all the page content was loaded:
  ```
  window.onload = function () {
        document.getElementById(“theId”).innerHTML = “New text”;
  }
  ```

3. Now we need to create some content for the main tag. Create variables and values for two units:

- The name of the first unit: Mathematics
- The grade for that unit: 7.5
- The name of the second unit: Literature
- The grade of that unit: 11.2

4. Set the content of the main tag to look like this (see next page for more information) :



Learning Platform

John Smith grades

Ma Mathematics

Grade: 7.5

Li Literature

Grade: 11.2

NOTE: Do not worry if you cannot reach the exact same result as long as you can create and show the new content in the window in a formatted way.

You will need:

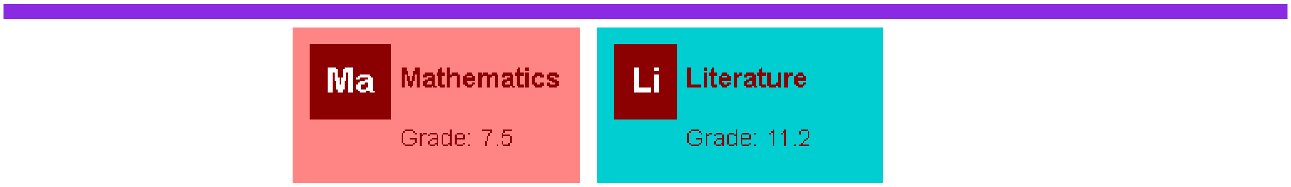
- To change the display of the main tag to flex in the studentGrades.css file
- To create elements for each unit filled with the information on the variables you created
  - You can use sections for each unit
  - You can obtain the 2 first letters using the substring method of strings:
    - https://www.w3schools.com/jsref/jsref_substring.asp
- To create the CSS for those elements (color, background-color, width, etc). You can create a class for them if you want.

Remember:

When you set the innerHTML property the text you use is interpreted as HTML, so recreate the all page if you want. You can create an auxiliary variable using **let** and concatenate the HTML you want piece by piece:

```
var text = “My text”;
var num1 = 12;
var num2 = 5;
window.onload = function () {
   let aux = “”;
   aux += “<h1>” + text + ”</h1>”;
   aux += “<p class=’a’> Sum = ” + (num1+num2) + “ </p>”;
   document.getElementById(“theId”).innerHTML = aux;
   // the element will now contain the HTML: <h1>My text</h1><p class=’a’> Sum = 17 </p>
}
```

5. We will now use a different background color if the student has failed the unit. You should use a conditional to check if the grade is below 9.5 for each unit. If the grade is below 9.5 you use a “failed” class in the html of that unit. In the CSS set the background color for that class. The result should look like this.



Ma Mathematics

Grade: 7.5

Li Literature

Grade: 11.2

(More information on the next page)

Remember:

The if conditional evaluates a boolean expression and runs the following block if it is true. If there is an else clause the else block will be run only if the Boolean expression evaluates to false.

You can create different string combinations using the conditional. There are several ways of using the conditional, here are two examples:

```
var num = 3;
window.onload = function () {
      let html = “”;
      if (num > 5) {
         html += “<h1 class=’red’> My text </h1>”;
      } else {
         html += “<h1> My text </h1>”;
      }
      document.getElementById(“theId”).innerHTML = html;
}
```

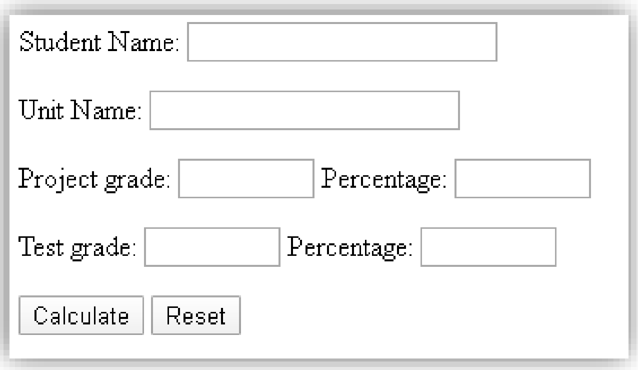
In the example above we use the then and else clauses to decide which text to use (we could remove the else clause if nothing was to be shown if the value was bellow or equal to 5).

```
var num = 5;
window.onload = function () {
   let aux = “”;
   if (num > 5) {
       aux = ” class=’red’”;
   }
   html += “<h1” + aux + ”> My text </h1>”;
   document.getElementById(“theId”).innerHTML = html;
}
```

In the example above the result is the same, the string with the class will be included in the string if the number is above 5, if not the aux variable will be the empty string and will not change the default text.

6. We will now create a form in the grading.html that we created in the HTML guide (if you do not have this file create it). We will not need a CSS file for now, but you can create one if you want. You also use the style.css to obtain the web site style.
   Create a grading.js file in the javascript directory and connect it to the html file.

7. Create the inputs for the following information:
    - Name of the unit to grade
    - Name of the student to grade
    - Project grade
    - Project percentage (ex: 0.6)
    - Test grade
    - Test percentage
    - A button to calculate the grade
    - A button to reset all the elements



The form will look like the image on the side. Make sure you follow the good practices for forms:

- Use the correct input type (text, number, password, date, color, button, etc)
- When we click on the text relative to the input the browser should focus on the input (use the label tag)
- For numeric values be sure to define the minimum (min property), maximum (max property) and the increment (step property).

8. Below the buttons create a paragraph tag with an id where we will write our results (it is empty for now).

9. Each button should have an onclick event that calls a function that we will define on the javascript file. Create this event handlers and the corresponding functions on the javascript file to provide the following functionality:
    - The reset button will delete all input values and the result paragraph content
    - The calculate button will obtain all the values, calculate the grade and insert the result on the paragraph tag.
    - The result should follow the format of this example:

      ```
      Student John Smith obtained 11.3 on the Mathematic unit.
      ```

        - Make sure you only show 1 number after the floating point.
        - Use auxiliary variables to hold the values and any calculations you need.
        - You may want to check if the user really filled the input or not i.e. if the obtained value is “” (empty string).
        - You can also check if the sum of the percentages is 1.

(More information on the next page)

Remember:

- You can create an event by setting the onclick property of the tag with a call to a javascript function

```
 <input type="button" value="Finish" onclick="finish()">
```

- Javascript functions are easy to declare, in this case the function does not receive any parameters and has no return value (it is a procedure):

```
function finish() {
    document.getElementById("end").innerHTML = "The End";
}
```

- You can obtain or change the value of one input by using the property value:
    - Change the value of input with id 'name' to empty:

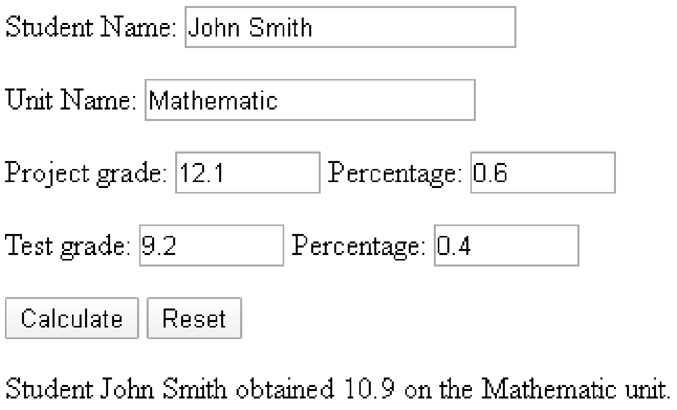
```
    document.getElementById("name").value = ""; //input will now be empty => reset
```

    - Obtain the value of input with id 'name' and use that value to say hello in the tag with id 'say'.

```
    let name =  document.getElementById("name").value;
    document.getElementById("say").innerHTML = "Hello "+name;
```

- All the values obtained from the input property value will be text (you can use valueAsNumber property for numbers, it will be null if it is not a number) but javascript converts a text number to number when you use numeric operators so you probably not notice the difference (you can also parse the text using parseFloat)

An image of the result:



Student Name: John Smith

Unit Name: Mathematic

Project grade: 12.1 Percentage: 0.6

Test grade: 9.2 Percentage: 0.4

Student John Smith obtained 10.9 on the Mathematic unit.

10. The form is finished, you can now change it to obtain a better-looking interface.

You can also improve the code made for the **5.** Step by creating a function (always avoid code repetition when possible).