To: Khattar daou

Ali nehme, Dmitry kizyakov, rami chaouki

john abbott college

Full Stack Web Development

Sunny side of the moon website

user interface integration project

Contents

[Case 1: 2](#_Toc103802056)

[Creating twice the number of pages as we have teammates. 2](#_Toc103802057)

[Case 2: 2](#_Toc103802058)

[Using *two* functions to enhance the user interface. 2](#_Toc103802059)

[Case 3: 3](#_Toc103802060)

[Using *arrays* and *loops* to enhance code. 3](#_Toc103802061)

[Case 4: 3](#_Toc103802062)

[Create a DOM tree of an HTML page 3](#_Toc103802063)

[Case 5: 0](#_Toc103802064)

[Apply validation to at least three of the form controls. 0](#_Toc103802065)

[Case 6: 0](#_Toc103802066)

[Allow for users to modify the input fields of a form by using a selector. 0](#_Toc103802067)

[Case 7: 1](#_Toc103802068)

[Use local storage to transfer information page to another 1](#_Toc103802069)

[Case 8: 1](#_Toc103802070)

[Have the group demonstrate enhancements and showcase their work to each other. 1](#_Toc103802071)

[Case 9-a: 2](#_Toc103802072)

[Implement the use of AJAX 2](#_Toc103802073)

[Case 9-b: 3](#_Toc103802074)

[Implement AJAX calls to communicate with an API 3](#_Toc103802075)

[Case 10: 3](#_Toc103802076)

[Use jQuery in your functions 3](#_Toc103802077)

# Case 1:

## Creating twice the number of pages as we have teammates.

This requirement has been met since our team contains 3 member and our website has 18 pages.

# Case 2:

## Using *two* functions to enhance the user interface.

The first part of this requirement was met with Ali’s *changeLang()* function, located in **splash.js**, which used DOM manipulation to toggle the language without the inconvenience of refreshing the page and restarting the animation.

The second part of this requirement was met with Ali’s *addInfo()* function located in **formB.js** which cleverly allows us to add new or remove extra adult fields selected by the user through DOM manipulation.

# Case 3:

## Using *arrays* and *loops* to enhance code.

Using arrays and loops allows us to create scalable code that can stay robust even when new code is added.

A great example of this is Ali’s *formFillValidator()* in **formB.js**. The function contains two loops, the first one cycles through all the input fields and adds to an array all the empty objects. The second loop goes through the array of empty objects and appends to them a warning message on submission. If more fields were added in the future, the code validating the empty fields would not need to be changed!

Dmitry also used this method in *readAdditionalPassengers()* in **localStorage.js** to loop through input fields of adults and children to populate an array with objects which can then be converted to JSON.

# Case 4:

## Create a DOM tree of an HTML page

For this case, Dmitry took a bullet for the team and did a great job mapping out the DOM of our index.html page. (see following page)

![Diagram, engineering drawing

Description automatically generated]()

# Case 5:

## Apply validation to at least three of the form controls.

This was accomplished by Ali in **formB.js** with the use of several helper functions like *alphaTextValidator(inputList)* that implemented **regex** to ensure that the fields entered were conforming to what was expected.

# Case 6:

## Allow for users to modify the input fields of a form by using a selector.

This is exactly what we set out to accomplish with the *addInfo()* function in **formB.js** which actually takes a dom selector element as a parameter and uses the option chosen by the user to create the requested amount of fields.

# Case 7:

## Use local storage to transfer information page to another

This was the work of Dmitry, who ensured that the name and number of adults would transfer from the **formB.html**file to the **successful-process.html** page. The **localStorage.js** contains the script where he stringifies the javascript object literal into a JSON object and stores it in localStorage and then use *window.location.assign()* function to redirect to **successful-process.html** and parse JSON and retrieve the data.

# Case 8:

## Have the group demonstrate enhancements and showcase their work to each other.

This was done throughout our group project as we were forced to communicate and review each other’s git merge requests. Dmitry and Ali were able to work on different aspects of formB.html and formB.js precisely because of the use of the content management system. Unlike our previous project, we did not allow pushing directly to origin main. This helped better safeguard the integrity of our project and forced us to communicate changes with each other.

# Case 9-a:

## Implement the use of AJAX

Rami took the lead on integrating Ajax to the project. In a bid to reduce the redundancy of having to change all html files when the nav bar was updated, he created **nav.js** which contained an AJAX call to the nav template **html-template.txt**. A secondary function, *setLang(en,fr)* was then called to fill in the parts of the template once it had loaded. The problem however, was that the function would fire before the AJAX call had been resolved. To overcome this hurdle, Rami made use of a promise, so that the AJAX call could promise that only upon its completion *.then* it would run the *setLang(en,fr)* function. The **nav.js** function was only implemented in FAQ.html and FAQ-fr.html because Rami noticed it performed worse than having the html directly embedded. It does remain however a functioning proof of concept for modularity.

# Case 9-b:

## Implement AJAX calls to communicate with an API

Rami’s second AJAX implementation touches on the FAQ.html, where he uses AJAX POST calls to submit the info directly to a node.js server run on Heroku. This method implements server-side validation than can be customized to return the status you want.

In fact, if you write in the English comment “Can you brew coffee?” it will respond with a status 418 “I am a teapot”.

Moreover, using the SendGrid API, which required a secret key in a .env folder. Each submission sends a customized template in either English or French depending on the page the form was submitted in.

# Case 10:

## Use jQuery in your functions

jQuery was used in most javascript files, primarily as a way of more efficiently accessing DOM objects. It can be found in **formB.js** and **nav.js** amongst other files.