Note taking web application

Paul Băltărete

Department of Computer Science, West University of Timișoara

January 2020

Contents

1 Motivation

The reason for building a note taking web application is that I find it useful to write down the things that you have to do or you just want to remember later. This simple thing, in my opinion, gives you productivity boost.

2 Technologies used

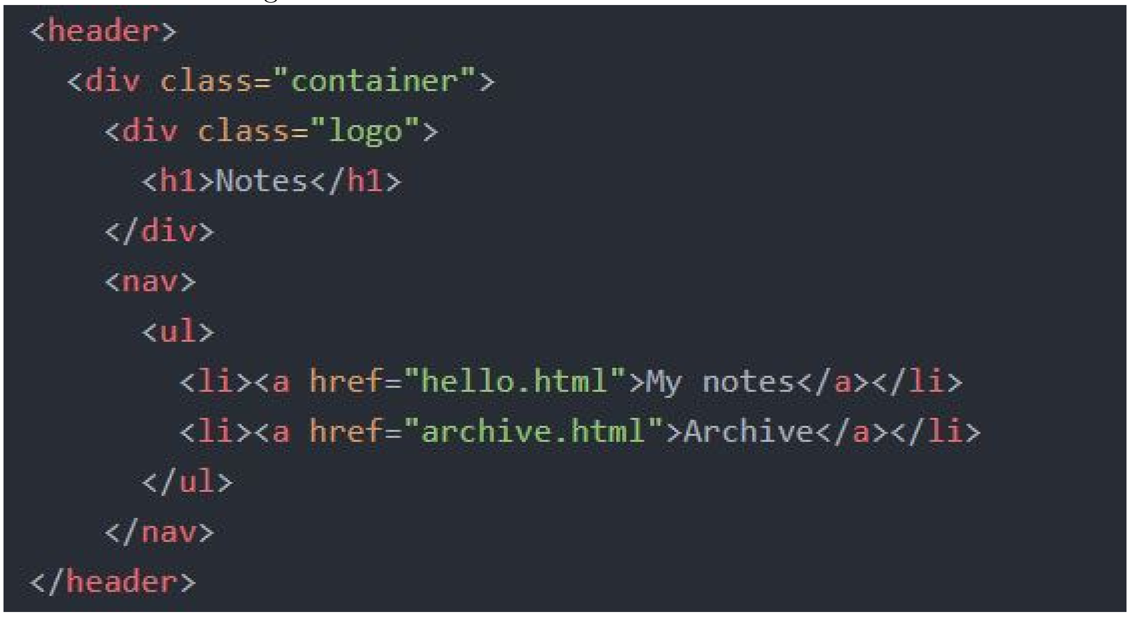
This section contains information about the technologies used for designing and implementing the functionality of the website.

HTML is used to describe the main layout of the website, CSS for wrapping the HTML elements into something beautiful and JavaScript together with JQuery for achieving the desired functionality of the website.

2.1 HTML (HyperText Markup Language)

The website is composed of two HTML files. One of them creates the home page or the "My Notes" page and the second one is used for creating the "Archive" page.

This is how the navigation menu of the website is created:



```
<header>
  <div class="container">
    <div class="logo">
      <h1>Notes</h1>
    </div>
    <nav>
      <ul>
        <li><a href="hello.html">My notes</a></li>
        <li><a href="archive.html">Archive</a></li>
      </ul>
    </nav>
</header>
```

In the picture above you can see the two available options of the navigation menu: "My notes" which is linked to the "hello.html" and "Archive", which is linked to the "archive.html" file.

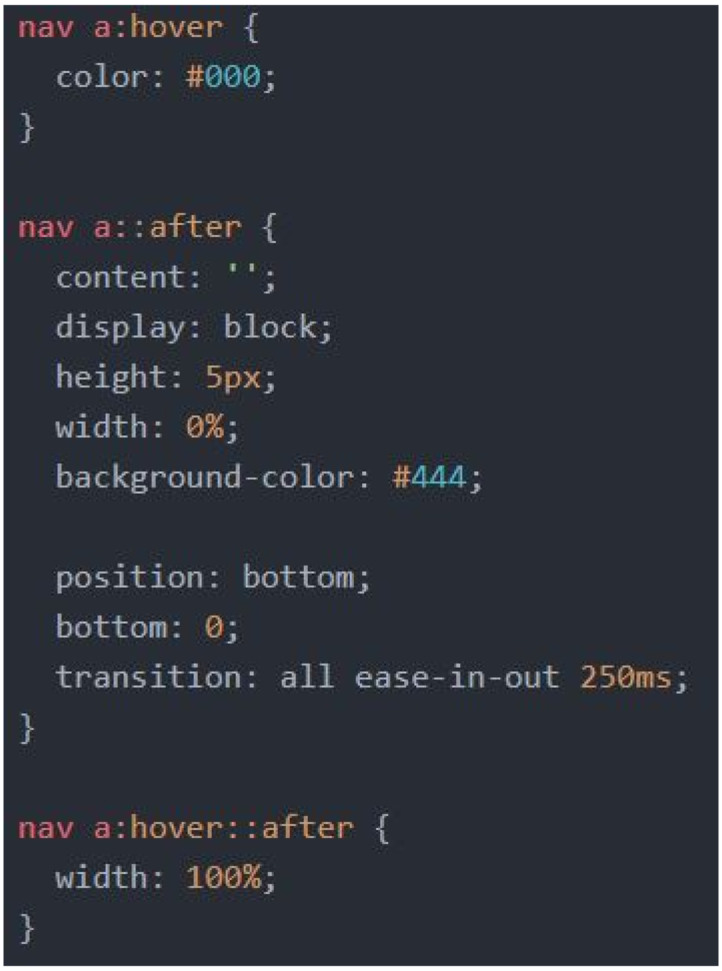
Every new note created will be attached to a *div* element with id "mainDIV" or "archiveDiv" using JavaScript. This *div* element is present in both of the HTML files, having a different id but belonging to the same class.

The HTML code is similar for both of the pages, except for one JavaScript file encapsulated between $< script >< /script >$ tags at the bottom of each HTML file and the id of the main div element.

2.2 CSS (Cascading Style Sheets)

The project contains three CSS files: one for styling the navigation menu and the page itself, one for the input box style and one for styling the notes.

Each of the items in the navigation menu has an effect attached to it. This effect starts when the mouse is hovering over one of the menu items. Both of the items contain a very small rectangle underneath them (or $after$ them). When hovering over one of the items, the width of the invisible rectangle starts increasing in size. The color of the menu item changes on hovering as well.

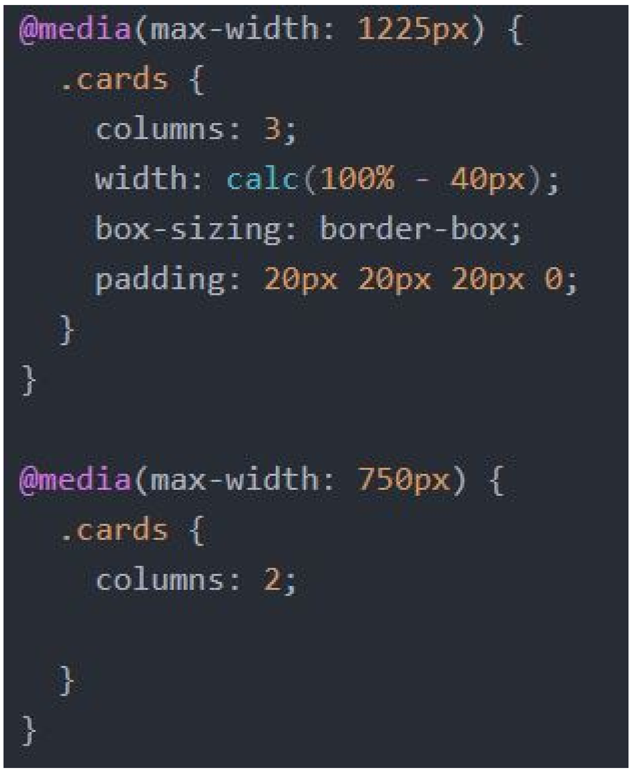


```
nav a:hover {
  color: #000;
}

nav a::after {
  content: '';
  display: block;
  height: 5px;
  width: 0%;
  background-color: #444;

  position: bottom;
  bottom: 0;
  transition: all ease-in-out 250ms;
}

nav a:hover::after {
  width: 100%;
}
```

By using $::\ hover$ and $::\ after$ pseudo-elements I could achieve the desired result.

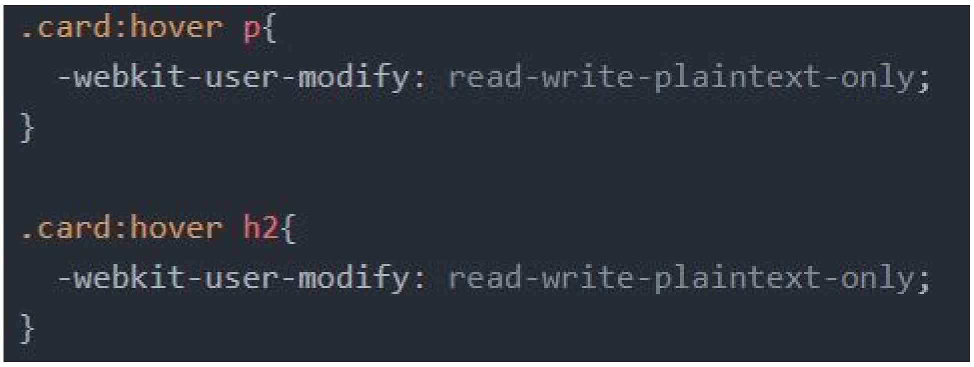
The size of rectangle increases gradually. This is achieved by using the $allease - in - out$ transition. This way, the rectangle will increase its width over the course of 250ms, until it reaches the width of the menu item.

All of the notes are arranged on 4 columns, having a fixed width and a flexible height. If the width of the window shrinks, the notes will be rearranged on 2 or 3 columns, depending on the width of the window.



```
@media(max-width: 1225px) {
  .cards {
    columns: 3;
    width: calc(100% - 40px);
    box-sizing: border-box;
    padding: 20px 20px 20px 0;
  }
}

@media(max-width: 750px) {
  .cards {
    columns: 2;

  }
}
```

I have used $::$ *hover* pseudo-element to make the title and the content of a specific note editable while hovering over it.



```
.card:hover p{
  -webkit-user-modify: read-write-plaintext-only;
}

.card:hover h2{
  -webkit-user-modify: read-write-plaintext-only;
}
```

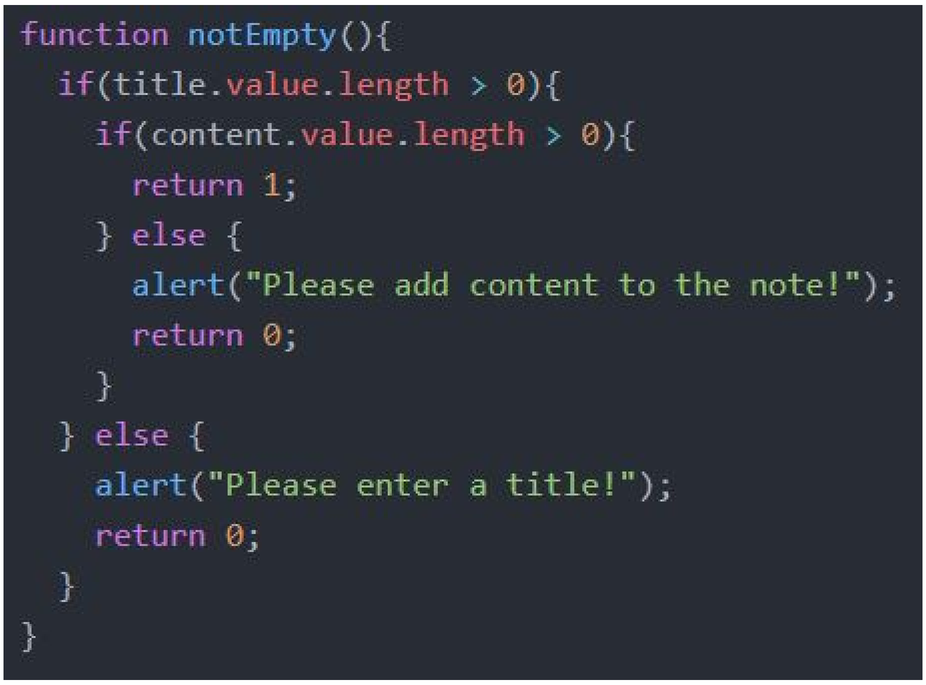
2.3 JavaScript

JavaScript has been used to implement the functionality of the site. The main operations are:

- adding a note
- deleting a note
- changing the color of a note
- archiving a note

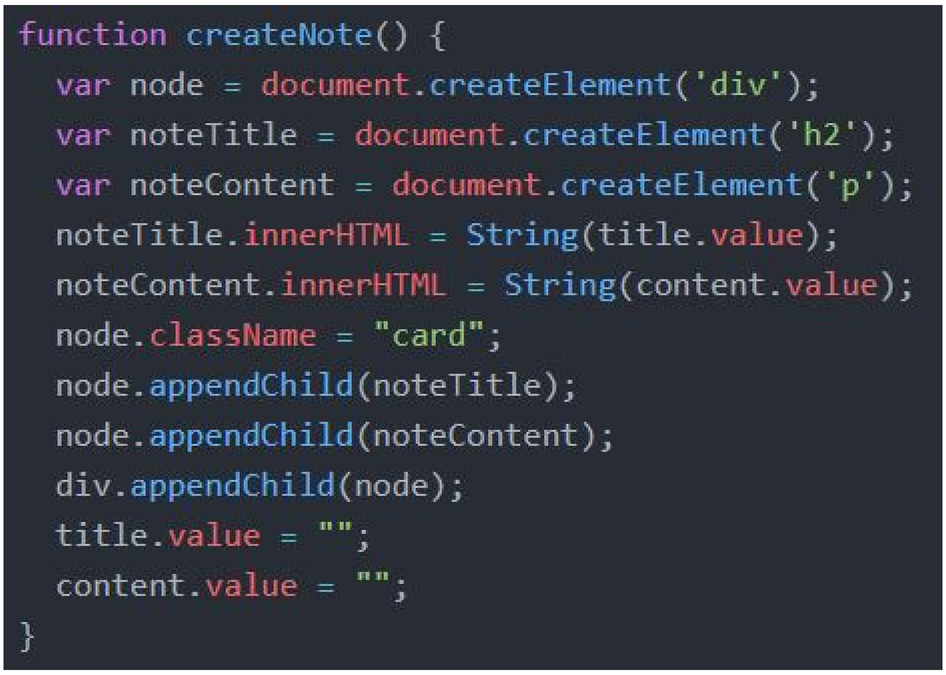
All of these operations are performed dynamically. This means that whenever you choose one of the options mentioned earlier, new HTML elements are added, removed or modified in some way in real time.

In order to add a note to ”My notes” you first have to add a title and some content to the note. If the user fails to add one of the two, a new event is triggered and an alert message pops at the top of the page. The following function does just that:



```
function notEmpty(){
  if(title.value.length > 0){
    if(content.value.length > 0){
      return 1;
    } else {
      alert("Please add content to the note!");
      return 0;
    }
  } else {
    alert("Please enter a title!");
    return 0;
  }
}
```

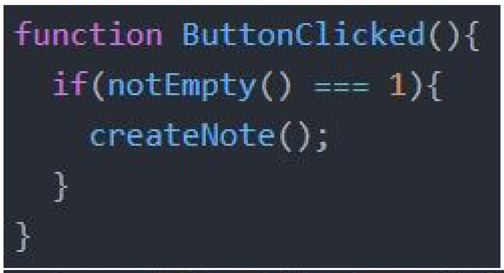
Once both of the text fields had been filled in, the user can press the + button and the note will be added to ”My notes”.



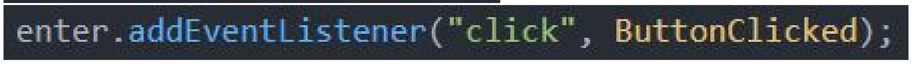
```
function createNote() {
  var node = document.createElement('div');
  var noteTitle = document.createElement('h2');
  var noteContent = document.createElement('p');
  noteTitle.innerHTML = String(title.value);
  noteContent.innerHTML = String(content.value);
  node.className = "card";
  node.appendChild(noteTitle);
  node.appendChild(noteContent);
  div.appendChild(node);
  title.value = "";
  content.value = "";
}
```

By creating a new *div* and attaching to it an *h*2 and a *p* element, a new note is created. The new div element is appended to the main div.

The *ButtonClicked* function is linked to the + button. This way, whenever the + button is clicked, the event is triggered.



```
function ButtonClicked(){
  if(notEmpty() === 1){
    createNote();
  }
}
```

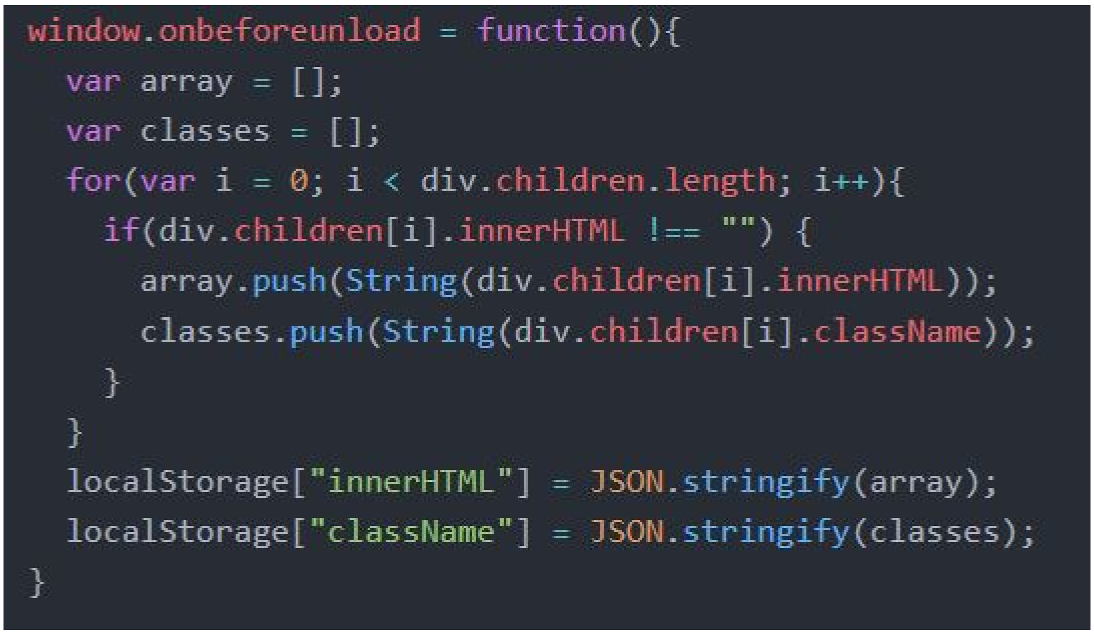


```
enter.addEventListener("click", ButtonClicked);
```

If the text fields are not empty, a new note is created by calling the *createNote* function.

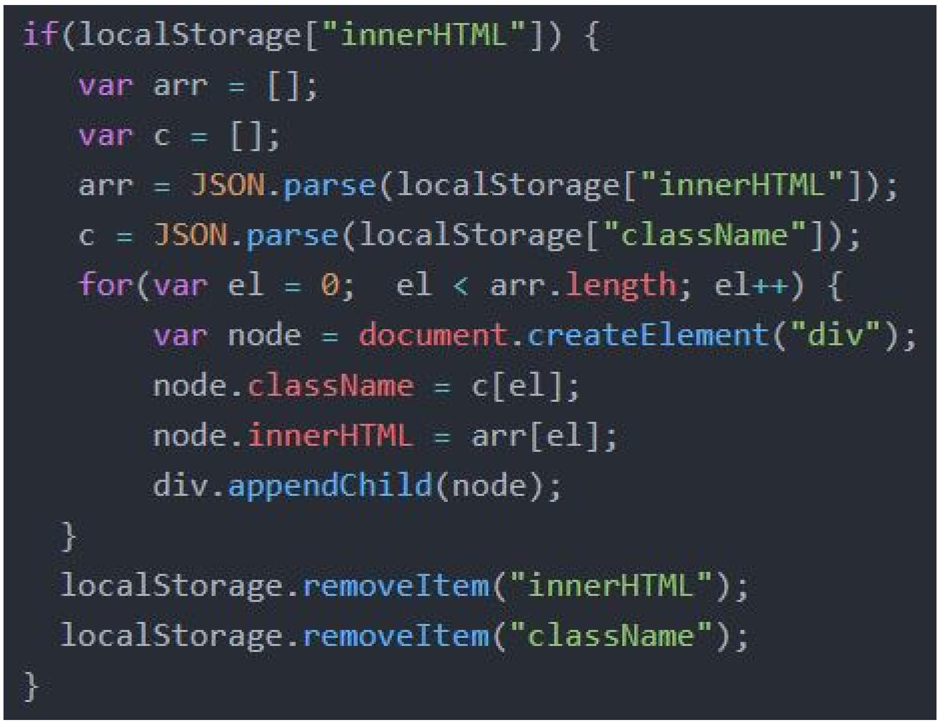
The new dynamically created web page is not saved locally, therefore *localStorage* is used to save all the new HTML elements created.

Using web storage, this website stores the data locally within the user's browser. This means that the notes are being saved even if the user refreshes or leaves the page.



```
window.onbeforeunload = function(){
  var array = [];
  var classes = [];
  for(var i = 0; i < div.children.length; i++){
    if(div.children[i].innerHTML !== "") {
      array.push(String(div.children[i].innerHTML));
      classes.push(String(div.children[i].className));
    }
  }
  localStorage["innerHTML"] = JSON.stringify(array);
  localStorage["className"] = JSON.stringify(classes);
}
```

This way, if the user returns to the website, all of their notes are preserved from the *localStorage*. We can achieve this by checking at the beginning of the JavaScript file if the *localStorage* is not empty.

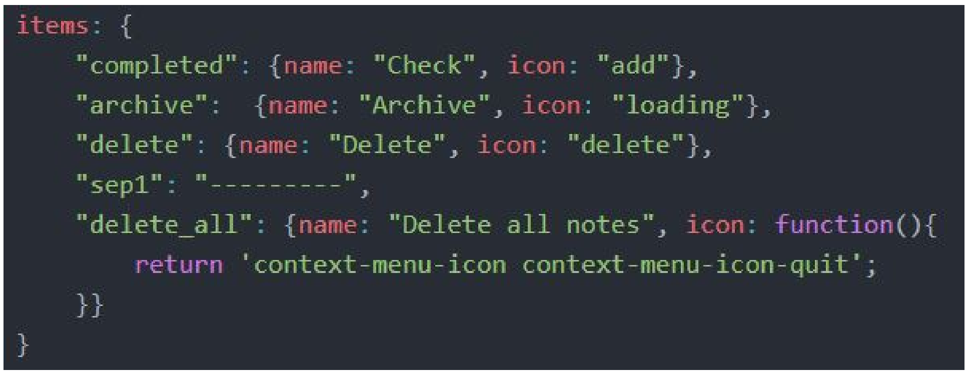


```
if(localStorage["innerHTML"]) {
   var arr = [];
   var c = [];
   arr = JSON.parse(localStorage["innerHTML"]);
   c = JSON.parse(localStorage["className"]);
   for(var el = 0;  el < arr.length; el++) {
       var node = document.createElement("div");
       node.className = c[el];
       node.innerHTML = arr[el];
       div.appendChild(node);
  }
  localStorage.removeItem("innerHTML");
  localStorage.removeItem("className");
}
```

2.4 JQuery

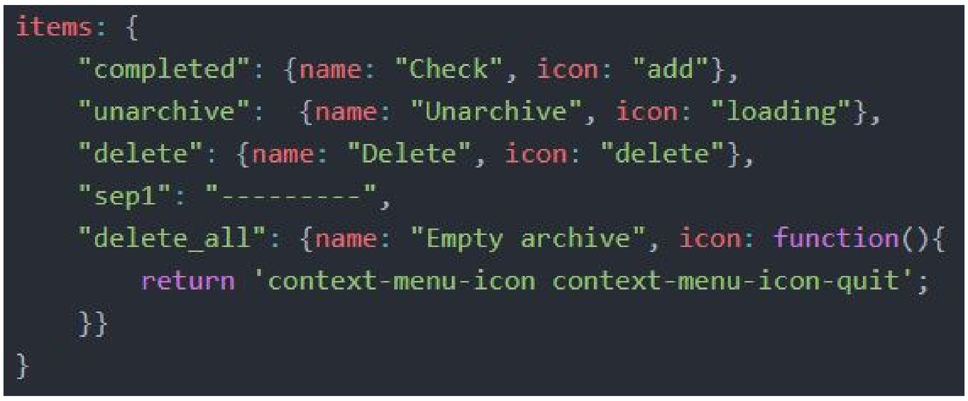
JQuery is a JavaScript library. This library simplifies JavaScript programming by introducing a new way of interacting with HTML elements.

With the help of JQuery the context menu of this website has been created.



```
items: {
    "completed": {name: "Check", icon: "add"},
    "archive":  {name: "Archive", icon: "loading"},
    "delete": {name: "Delete", icon: "delete"},
    "sep1": "---------",
    "delete_all": {name: "Delete all notes", icon: function(){
        return 'context-menu-icon context-menu-icon-quit';
    }}
}
```

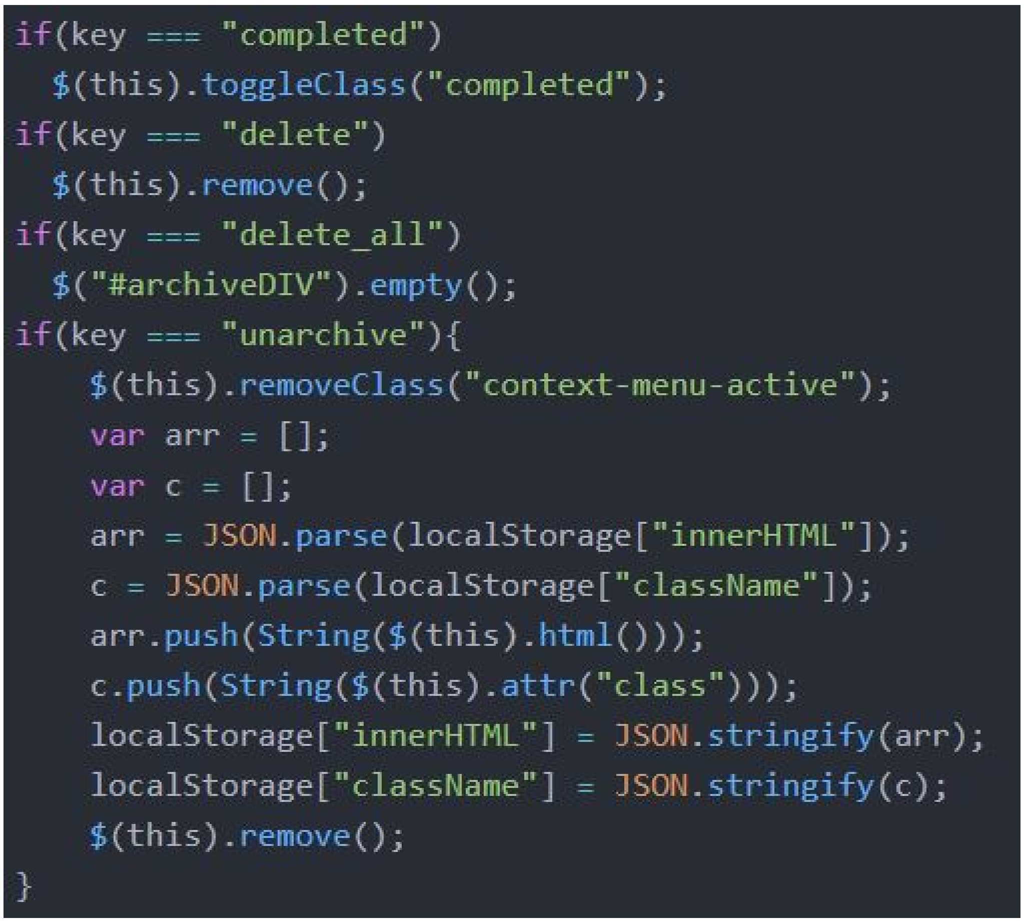
The context menu of the home page's notes



```
items: {
    "completed": {name: "Check", icon: "add"},
    "unarchive":  {name: "Unarchive", icon: "loading"},
    "delete": {name: "Delete", icon: "delete"},
    "sep1": "---------",
    "delete_all": {name: "Empty archive", icon: function(){
        return 'context-menu-icon context-menu-icon-quit';
    }}
}
```

The context menu of the archive page's notes

A series of if statements are used to check which of the context menu's items has been clicked. The picture below is a snippet of the code which represents the rationale behind the archive page's context menu.


```
if(key === "completed")
  $(this).toggleClass("completed");
if(key === "delete")
  $(this).remove();
if(key === "delete_all")
  $("#archiveDIV").empty();
if(key === "unarchive"){
    $(this).removeClass("context-menu-active");
    var arr = [];
    var c = [];
    arr = JSON.parse(localStorage["innerHTML"]);
    c = JSON.parse(localStorage["className"]);
    arr.push(String($(this).html()));
    c.push(String($(this).attr("class")));
    localStorage["innerHTML"] = JSON.stringify(arr);
    localStorage["className"] = JSON.stringify(c);
    $(this).remove();
}
```

3 Future work

An nice addition to the website would be a server with database linked to a sign in/up form where you can create your own account and be able to access all of your notes from any device with internet access. This would make the website more reliable and easy to use.

Being able to share the notes and have a common panel with the members of your team is an useful feature that can be implemented once the database is set up.

Glossary

database A database is an organized collection of data, generally stored and accessed electronically from a computer system. 10

div The $< div >$ tag defines a division or a section in an HTML document. 3, 4, 7

id In CSS the $\#id$ selector styles the element with the specified id. 3, 4

server In computing, a server is a computer program or a device that provides functionality for other programs or devices, called "clients". 10

Acronyms

CSS Cascading Style Sheets. 3, 4

HTML HyperText Markup Language. 2–4, 6, 7, 9