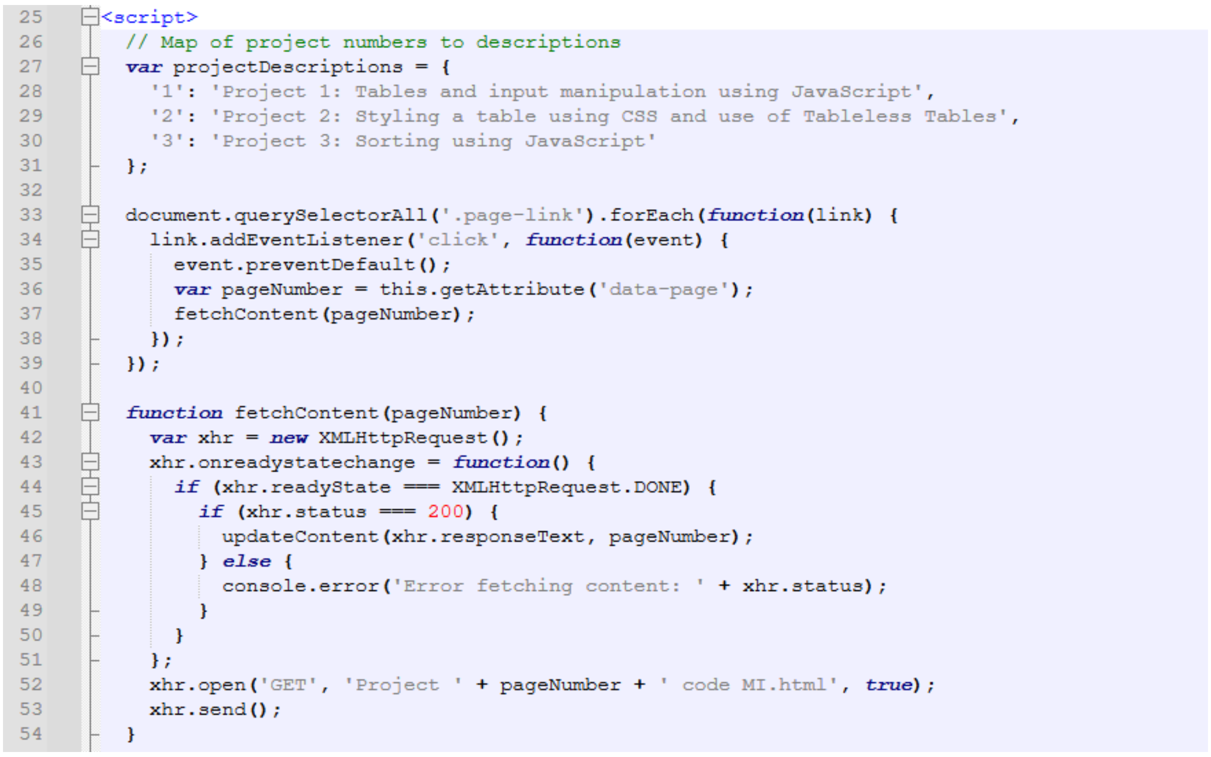
Warsaw University of Technology	Advanced Internet Programming
Project 5 AJAX Technology and its applications	Maciej Iwańczyk 311258
Date: 22.03.2024	Grade:

1. Introduction:

In today's project I had to recreate Project 4 using AJAX and create a smooth navigation menu for Projects no. 1-4 as it is done on this website: http://www.dynamicdrive.com/dynamicindex1/ddsmoothmenu.htm

2. Update of Project 4 into AJAX:

In order to update my Project 4 with the usage of AJAX I only had to change the JavaScript part of the program. I have started with writing the fetchContent(pageNumber) function that is responsible for making an AJAX request to fetch the content of the selected page.
It creates a new XMLHttpRequest object xhr and sets up an event handler for when the state changes. Upon receiving a successful response (status code 200), it calls the updateContent() function with the retrieved content and page number. If an error occurs during the request, it logs an error message to the console. Thus AJAX implementation in the project.

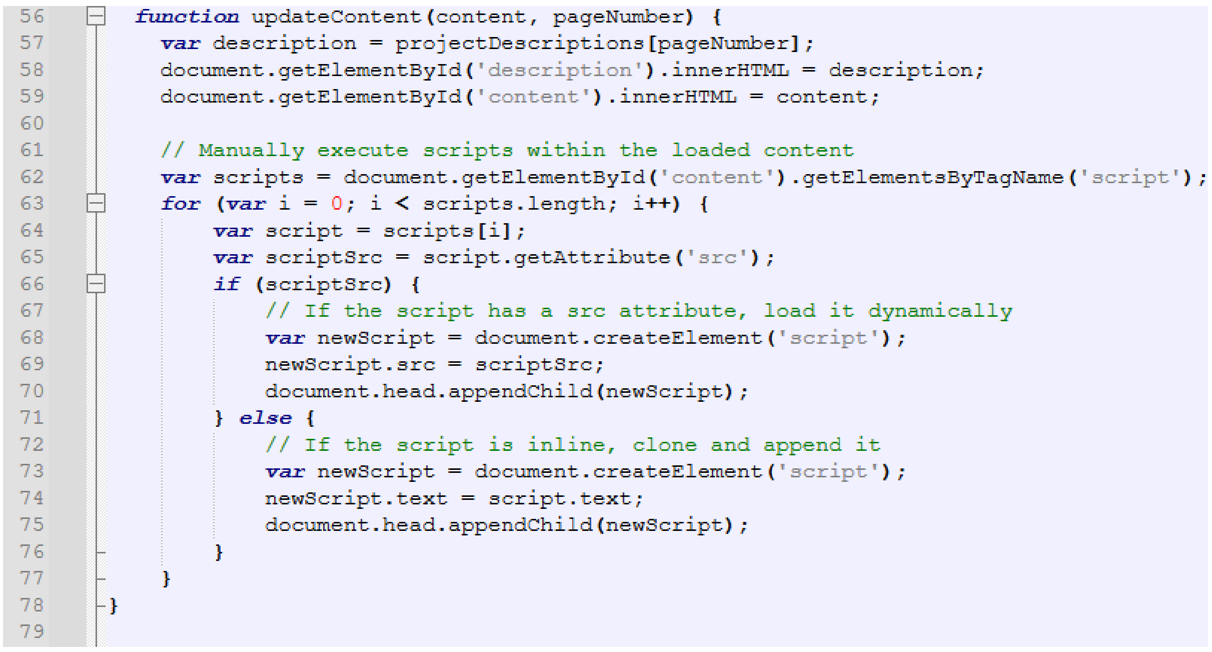


```
<script>
  // Map of project numbers to descriptions
  var projectDescriptions = {
    '1': 'Project 1: Tables and input manipulation using JavaScript',
    '2': 'Project 2: Styling a table using CSS and use of Tableless Tables',
    '3': 'Project 3: Sorting using JavaScript'
  };

  document.querySelectorAll('.page-link').forEach(function(link) {
    link.addEventListener('click', function(event) {
      event.preventDefault();
      var pageNumber = this.getAttribute('data-page');
      fetchContent(pageNumber);
    });
  });

  function fetchContent(pageNumber) {
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {
      if (xhr.readyState === XMLHttpRequest.DONE) {
        if (xhr.status === 200) {
          updateContent(xhr.responseText, pageNumber);
        } else {
          console.error('Error fetching content: ' + xhr.status);
        }
      }
    };
    xhr.open('GET', 'Project ' + pageNumber + ' code MI.html', true);
    xhr.send();
  }
```

However with this function came a big problem of not being able to run functions from Projects 1, 2 and 3. I have solved this problem by modifying the updateContent(content, pageNumber) function which previously I have made only to change the descriptions of the selected projects from within the Project 4 file.



```
function updateContent(content, pageNumber) {
  var description = projectDescriptions[pageNumber];
  document.getElementById('description').innerHTML = description;
  document.getElementById('content').innerHTML = content;

  // Manually execute scripts within the loaded content
  var scripts = document.getElementById('content').getElementsByTagName('script');
  for (var i = 0; i < scripts.length; i++) {
      var script = scripts[i];
      var scriptSrc = script.getAttribute('src');
      if (scriptSrc) {
          // If the script has a src attribute, load it dynamically
          var newScript = document.createElement('script');
          newScript.src = scriptSrc;
          document.head.appendChild(newScript);
      } else {
          // If the script is inline, clone and append it
          var newScript = document.createElement('script');
          newScript.text = script.text;
          document.head.appendChild(newScript);
      }
  }
}
```

updateContent(content, pageNumber) function updates the description and content divs with the fetched content. It retrieves the description for the selected page from the projectDescriptions object and updates the description div accordingly and it sets the innerHTML of the content div to the fetched content.

It also handles the execution of scripts within the loaded content which fixed my previous issue with not working javascripts:

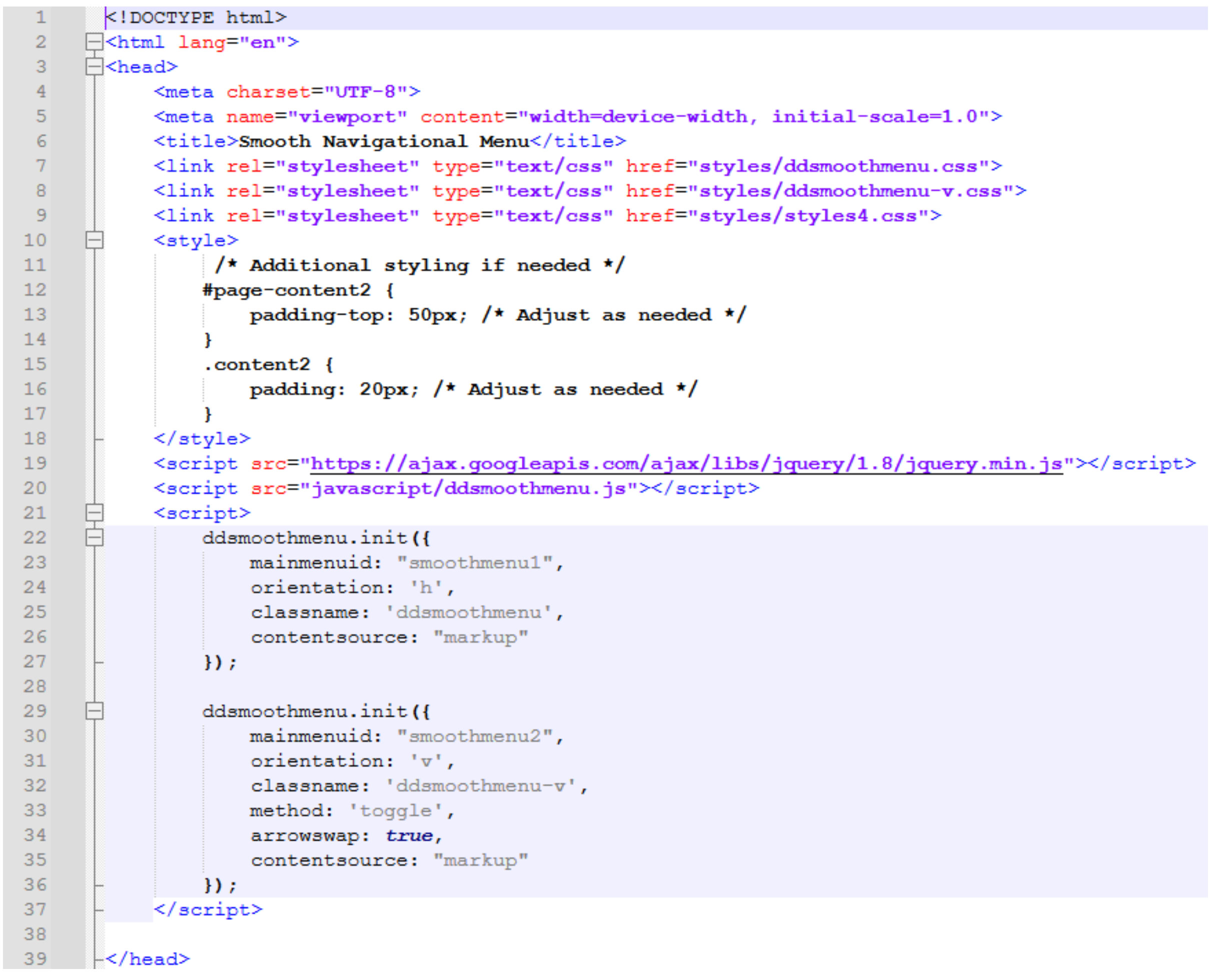
- It iterates over all script tags within the content div.
- For each script tag, it checks if it has a src attribute. If it does, it dynamically loads the script by creating a new script element and appending it to the document head with the src attribute set.
- If the script is inline (no src attribute), it clones the script element and appends it to the document head with its text content.

After those changes the page works correctly and AJAX was implemented.

3.Smooth Navigational Menu (v3.0):

I have used javascript and css files available from the website:

- ddsmoothmenu.js
- ddsmoothmenu.css
- ddsmoothmenu-v.css
- https://ajax.googleapis.com/ajax/libs/jquery/1.8/jquery.min.js (I had to add "s" to the https because with only http like it was on the tutorial website my project did not want to load the script after hosting it publicly because of security reasons.)

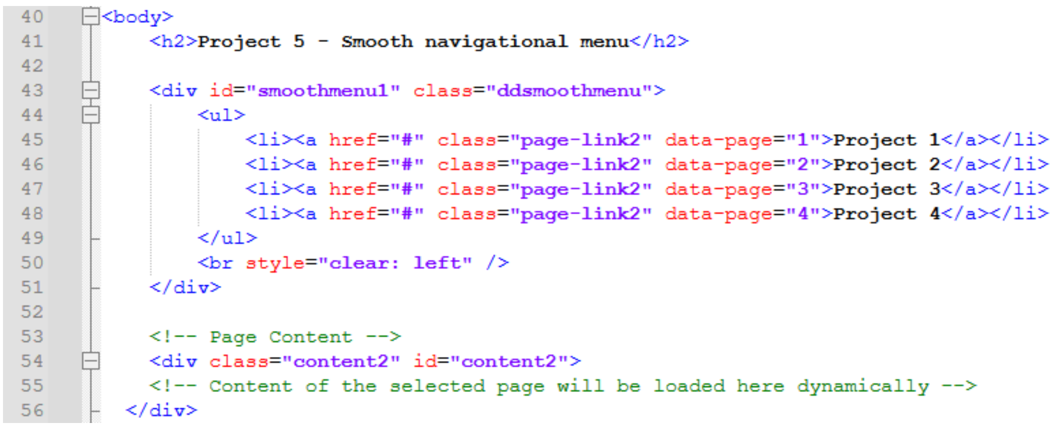


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Smooth Navigational Menu</title>
    <link rel="stylesheet" type="text/css" href="styles/ddsmoothmenu.css">
    <link rel="stylesheet" type="text/css" href="styles/ddsmoothmenu-v.css">
    <link rel="stylesheet" type="text/css" href="styles/styles4.css">
    <style>
         /* Additional styling if needed */
        #page-content2 {
            padding-top: 50px; /* Adjust as needed */
        }
        .content2 {
            padding: 20px; /* Adjust as needed */
        }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8/jquery.min.js"></script>
    <script src="javascript/ddsmoothmenu.js"></script>
    <script>
        ddsmoothmenu.init({
            mainmenuid: "smoothmenu1",
            orientation: 'h',
            classname: 'ddsmoothmenu',
            contentsource: "markup"
        });

        ddsmoothmenu.init({
            mainmenuid: "smoothmenu2",
            orientation: 'v',
            classname: 'ddsmoothmenu-v',
            method: 'toggle',
            arrowswap: true,
            contentsource: "markup"
        });
    </script>

</head>
```

Inside <script> tags, the ddsmoothmenu.init() function is called twice. The first call initializes a horizontal menu (orientation: 'h') with the id smoothmenu1. The second call initializes a vertical menu (orientation: 'v') with the id smoothmenu2. Both calls specify the class name for the menu's outer DIV, and they indicate that the menu contents are provided directly in the HTML markup (contentsource: "markup").

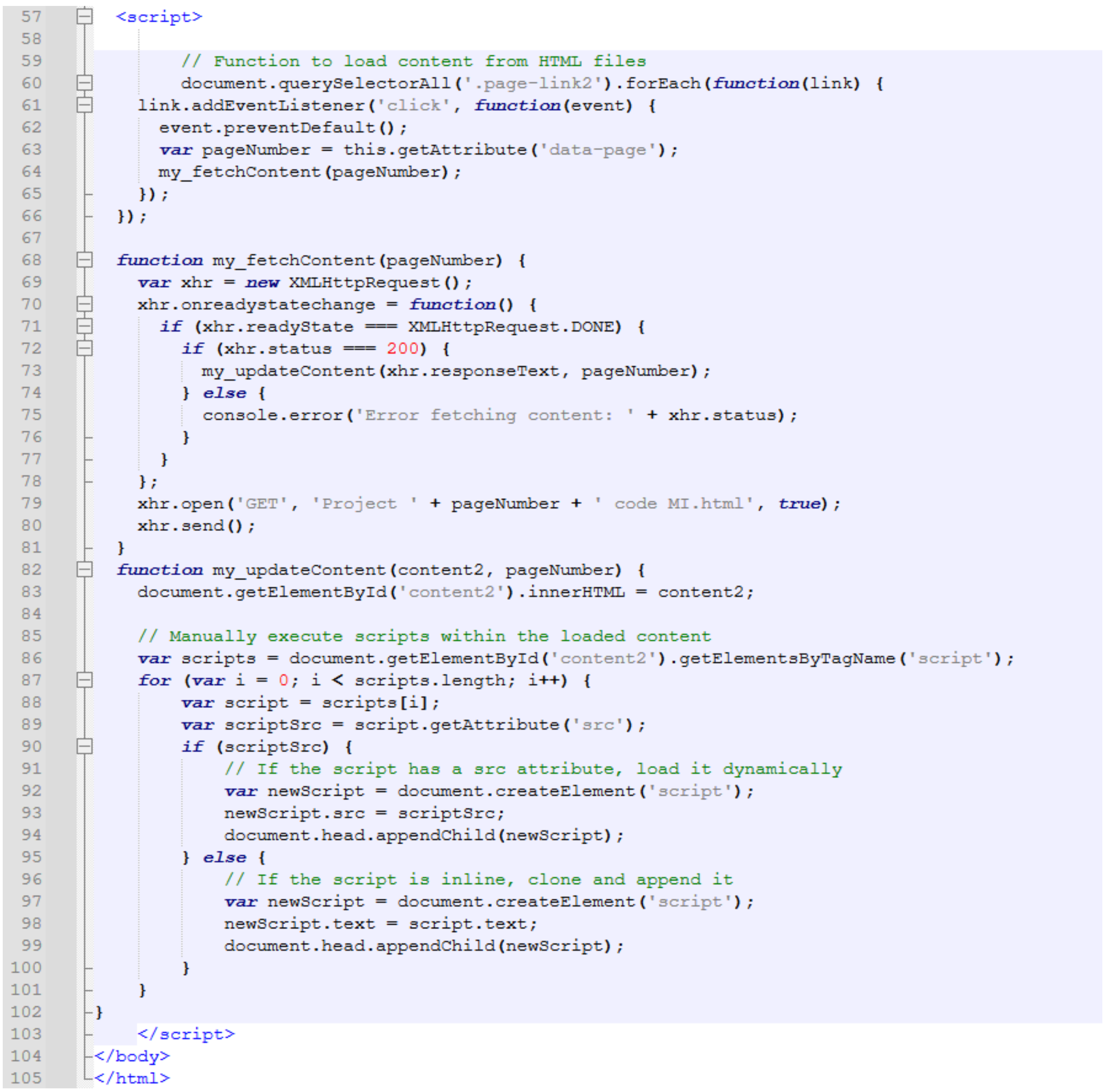


```
<body>
    <h2>Project 5 - Smooth navigational menu</h2>

    <div id="smoothmenu1" class="ddsmoothmenu">
        <ul>
            <li><a href="#" class="page-link2" data-page="1">Project 1</a></li>
            <li><a href="#" class="page-link2" data-page="2">Project 2</a></li>
            <li><a href="#" class="page-link2" data-page="3">Project 3</a></li>
            <li><a href="#" class="page-link2" data-page="4">Project 4</a></li>
        </ul>
        <br style="clear: left" />
    </div>

    <!-- Page Content -->
    <div class="content2" id="content2">
    <!-- Content of the selected page will be loaded here dynamically -->
  </div>
```

There's a hidden anchor tag <a> with the id ddsmoothmenu-mobiletoggle, which is used for toggling the mobile menu. A <div> element with the id smoothmenu1, containing an unordered list <ul> representing the main navigation menu. Each list item <li> contains a hyperlink <a> pointing to different project pages. I could also easily modify this menu to contain folders and sub items if I needed to.



```
  <script>

        // Function to load content from HTML files
        document.querySelectorAll('.page-link2').forEach(function(link) {
    link.addEventListener('click', function(event) {
      event.preventDefault();
      var pageNumber = this.getAttribute('data-page');
      my_fetchContent(pageNumber);
    });
  });

  function my_fetchContent(pageNumber) {
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {
      if (xhr.readyState === XMLHttpRequest.DONE) {
        if (xhr.status === 200) {
          my_updateContent(xhr.responseText, pageNumber);
        } else {
          console.error('Error fetching content: ' + xhr.status);
        }
      }
    };
    xhr.open('GET', 'Project ' + pageNumber + ' code MI.html', true);
    xhr.send();
  }
  function my_updateContent(content2, pageNumber) {
    document.getElementById('content2').innerHTML = content2;

    // Manually execute scripts within the loaded content
    var scripts = document.getElementById('content2').getElementsByTagName('script');
    for (var i = 0; i < scripts.length; i++) {
        var script = scripts[i];
        var scriptSrc = script.getAttribute('src');
        if (scriptSrc) {
            // If the script has a src attribute, load it dynamically
            var newScript = document.createElement('script');
            newScript.src = scriptSrc;
            document.head.appendChild(newScript);
        } else {
            // If the script is inline, clone and append it
            var newScript = document.createElement('script');
            newScript.text = script.text;
            document.head.appendChild(newScript);
        }
    }
}
    </script>
</body>
</html>
```

I have used the AJAX script from the updated Project 4 but I have changed the names of the functions so that they do not interfere with each other.

3. The final project:

The webpage: https://zstarwarss.github.io/Project-5-Maciej-Iwanczyk/
The files: https://github.com/ZstarwarsS/Project-5-Maciej-Iwanczyk

I have hosted the page with all the files that can be viewed on GitHub because there are many files so sending them via email would be problematic.