Bachelor’s thesis

Bachelor of Engineering Information and Communications Technology

2023

Sami Wazni

DEVELOPMENT OF A REACT-BASED PORTFOLIO FOR FRONT-END USAGE

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Bachelor of Engineering, Information and Communications Technology

2023 | 65

Sami Wazni

DEVELOPMENT OF A REACT-BASED PORTFOLIO FOR FRONT-END USAGE

This thesis delves into the fundamentals of front-end Development, exploring HTML, CSS, JavaScript, and ReactJS, explaining foundational concepts, and showcasing the union between these technologies through practical code examples. These examples guide the construction of a ReactJS portfolio, providing hands-on experience in front-end React application development. Topics covered include React Routers, Hooks such as useRef and useEffect, Components, and the crucial Virtual DOM.

The primary goal of this thesis was to describe front-end skills and construct a React-based portfolio seamlessly integrating HTML, CSS, JavaScript, and ReactJS. Enhanced functionality, supported by EmailJS, enables seamless email integration for sending emails through the portfolio. The portfolio serves as a comprehensive showcase of personal skills, projects, and contact information.

The work carried out in this thesis is also documented as a seven-week diary providing insights into the week-to-week challenges encountered at an IT company.

In conclusion, this thesis provides a thorough understanding of foundational front-end concepts. Applying these principles results in the creation of a React portfolio that effectively presents skills, information, and contact details. Furthermore, the thesis highlights practical problem-solving within the realm of front-end development, offering valuable insights into the dynamic nature of real-world web development projects.

Keywords:

HTML, CSS, JavaScript, ReactJS, Front-end

Contents

Figures

Pictures

Tables

List of abbreviations

CSS	Cascading Style Sheets, used for styling web content.
DOM	Document Object Model, a programming interface for web documents.
HTML	Hypertext Markup Language, used for structuring web content.
JS	JavaScript, a programming language used for creating dynamic web content.
JSX	JavaScript XML, an optional syntax extension for JavaScript used with React.

1 Introduction

The front-end, also known as the Client side, is one component of the Full-stack development paradigm, and it has witnessed significant expansion with the continual emergence of new tools from various developers. Within the context of front-end development, it becomes crucial to discern why one should learn or utilize specific tools, given the vast array available. This thesis endeavours to provide a clear starting roadmap for front-end development, offering insights into how websites or web applications can be effectively built. (Dziallas, 2020).

The foundational tools to initiate this thesis are Chapter 2: HTML, responsible for structuring web content, and CSS, enhancing the visual appeal and responsiveness of sites. With Chapter 3: JavaScript, which adds dynamism to websites, enabling actions such as navigation, data handling, and user experience enhancements. As front-end development progresses, specialized tools emerge to expedite website creation, with Chapter 4: React being a prominent example. React, a JavaScript library, stands as a cornerstone in the development of contemporary web application interfaces, providing developers with a robust toolkit for creating responsive, dynamic, and user-friendly applications.

In Chapter 5 of this thesis, the culmination of acquired front-end development skills will be showcased through the creation of a portfolio using React. This practical application will incorporate principles learned throughout the thesis, integrating HTML, CSS, JavaScript, and ReactJS. The implementation of EmailJS will also be explored to enhance the portfolio's functionality by seamlessly integrating an email-sending feature. This hands-on project aims to exemplify the practical application of acquired skills, providing readers with a tangible demonstration of front-end development in action.

In Chapter 6, the focus will shift to problem-solving at “Draivi, a performance-oriented company with a diverse portfolio of online media platforms, primarily centred around online lending”. Aligned with Draivi's dynamic objectives, this thesis embarks on a comprehensive exploration detailed in a seven-week diary.

The diary recounts experiences in both front-end and Back-end development, offering an account of challenges faced and innovative solutions devised to overcome them. This seven-week serves as a practical reflection, shedding light on the dynamic nature of web development projects and providing insights into the real-world application of acquired skills.

2 What is front-end?

Front-end development, also referred to as client-side development, is responsible for shaping the user interface of a website or web application. It constitutes a crucial aspect of Full-Stack development, covering both the front-end and Back-end domains. The construction of visual and interactive elements in the user interface relies on technologies such as HTML, CSS, and JavaScript.

Moreover, within the realm of front-end development, a wide range of libraries and frameworks significantly aids developers in creating dynamic, responsive, and faster web applications. Prominent examples include React.js, Vue.js, Angular.js, and Next.js. These tools empower developers by providing efficient solutions for building sophisticated and user-friendly interfaces, thereby enhancing the overall performance and responsiveness of web applications.

Websites or applications function through the exchange of queries and responses, interacting with and retrieving data from the back-end host system. These interfaces facilitate user engagement with information systems. Notably, front-end systems possess constrained computational and business logic processing capabilities, relying heavily on the data structures and functions provided by the host system. (Dziallas, 2020).

In the development of a website or application, the front-end serves as the user-facing component, facilitating interaction with server functionalities for tasks such as data transmission and reception. This encompasses various user actions, such as completing forms or activating functions, which seamlessly prompt the processing and transfer of data to the server, shielding users from intricate back-end operations.

Furthermore, the front-end plays a pivotal role in the aesthetic presentation and functional responsiveness of the site or application. It is instrumental in creating an interface that enhances user experience, ensuring a visually appealing and smoothly interactive platform. (Jazayeri, 2007)

3 HTML and CSS

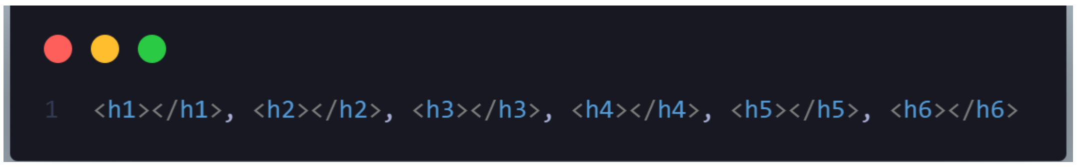
In this chapter, a brief overview of HTML and CSS fundamentals, including essential elements and operational principles, will be provided. By the end of the thesis, the knowledge required to build a portfolio website using these tools will be acquired.

3.1 Introduction to HTML

HTML is a Hypertext Markup Language used in creating Web pages. Each site must include the structure of the HTML because the browser wouldn't know how to interpret and display the information on the page if it did not include HTML. HTML provides the structural framework for a web page, defining elements such as headings, paragraphs, links, images, lists, forms, buttons, and inputs. (Hwangbo & Lee, 2008)

3.1.1 HTML Elements

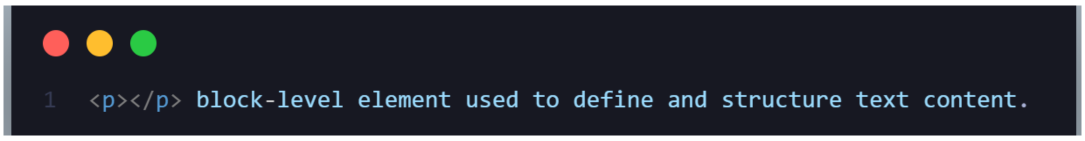
Headings: Each head has a font size that is already defined by default.



```
<h1></h1>, <h2></h2>, <h3></h3>, <h4></h4>, <h5></h5>, <h6></h6>
```

Figure 1. HTML Headings.

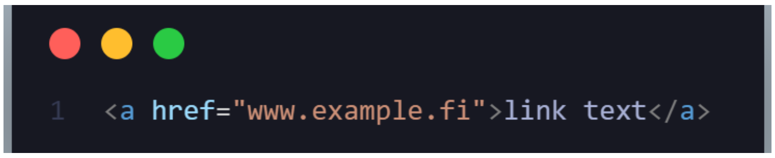
Paragraphs: The Paragraph has a font size that is already defined by default.



```
<p></p> block-level element used to define and structure text content.
```

Figure 2. HTML Paragraphs.

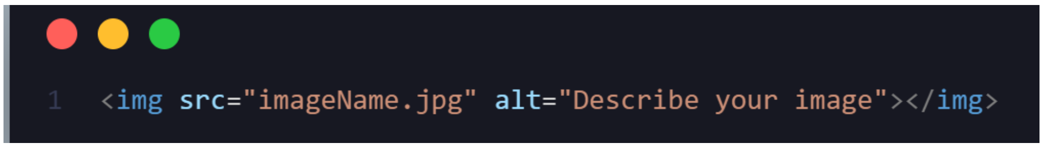
Links



```html
<a href="www.example.fi">link text</a>
```

Figure 3. HTML Links.

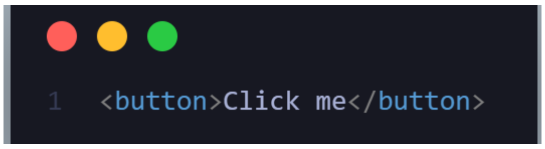
Images



```html
<img src="imageName.jpg" alt="Describe your image"></img>
```

Figure 4. HTML Images.

Button

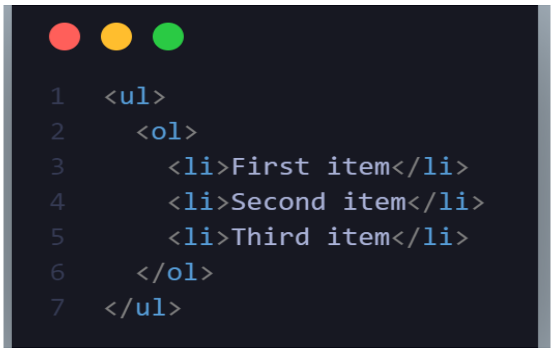


```html
<button>Click me</button>
```

Figure 5. HTML Button.

Lists

<ul></ul>: Ordered Lists <ol></ol> Each item in the list is preceded by a sequential number (1, 2, 3, etc.) by default.



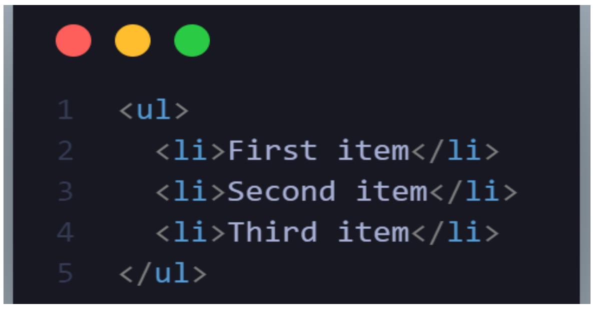
```html
<ul>
  <ol>
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
  </ol>
</ul>
```

Figure 6. HTML sequential number list.

The output:

- First item
- Second item
- Third item

Unordered Lists <ul></ul> are used to represent a list of items with no specific order or sequence. Each item in the list is typically preceded by a bullet point by default.



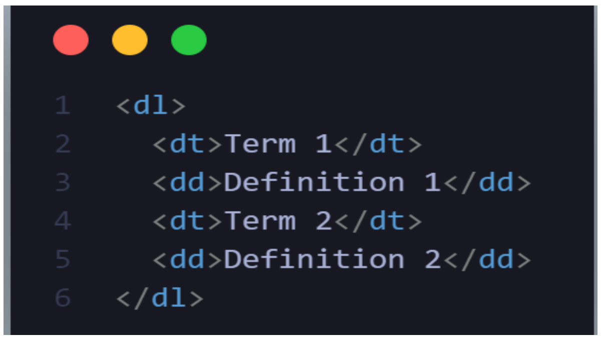
```
<ul>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ul>
```

Figure 7. HTML sequential bullet list.

The output:

- First item
- Second item
- Third item

Definition Lists <dl></dl> are used to define terms and provide their corresponding definitions. They consist of term-description pairs.



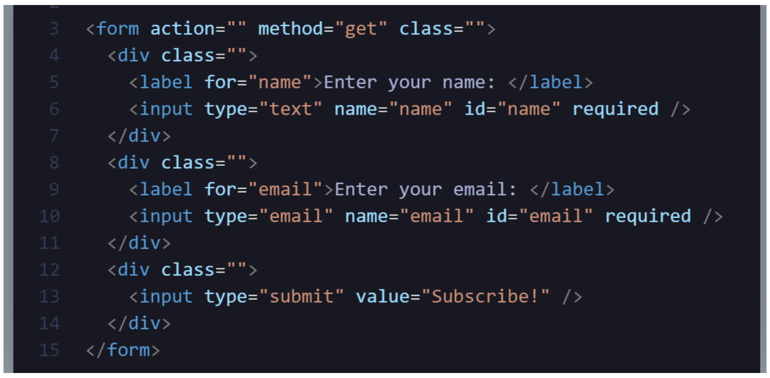
```
<dl>
  <dt>Term 1</dt>
  <dd>Definition 1</dd>
  <dt>Term 2</dt>
  <dd>Definition 2</dd>
</dl>
```

Figure 8. HTML sequential bullet list and sub-bullet.

The output:

- Term 1
    - Definition 1

Forms



```html
<form action="" method="get" class="">
  <div class="">
    <label for="name">Enter your name: </label>
    <input type="text" name="name" id="name" required />
  </div>
  <div class="">
    <label for="email">Enter your email: </label>
    <input type="email" name="email" id="email" required />
  </div>
  <div class="">
    <input type="submit" value="Subscribe!" />
  </div>
</form>
```

Figure 9. HTML Form.

There are many types of inputs in the HTML form. To specify the type of the input using type=" the type of the input". Below are the input types:

- type = "name": This is used for single-line text input. e.g. username, names.
- type = "password": Similar to the text input, but the entered text is masked for security. When the type is a password, it will automatically convert to asterisks "******".
- type = "radio": Allows the user to select one option from a list.
- type = "checkbox": Used for multiple-choice selections. Users can check multiple options.
- type = "number": Restricts input to numeric values and includes up and down arrows for adjustments.
- type = "email": Used for email addresses, with built-in validation.
- type = "date": Provides a date picker for selecting dates.
- type = "file": Allows users to upload files.

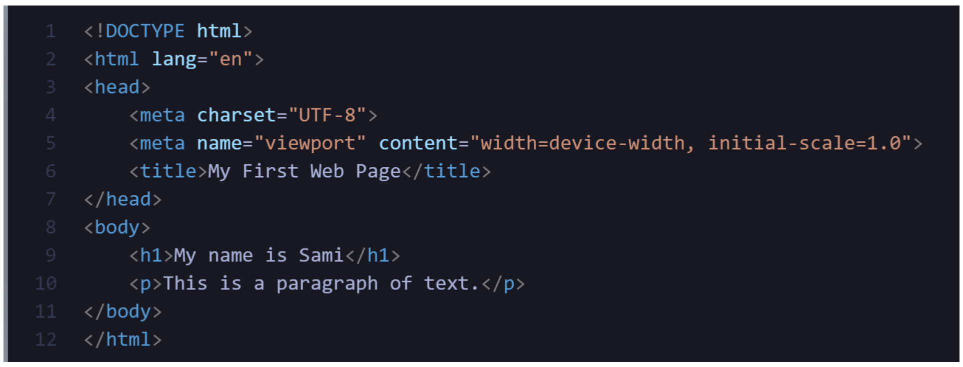
- type = “select”: Creates a dropdown menu for selecting one option from a list.
- type = “textarea”: Used for multi-line text input, such as comments or descriptions.

These input types are essential for creating interactive web forms that allow users to input various types of data. These can be customized further with attributes such as name, placeholder, and more to suit the specific needs.

Div and Section Elements: The <div> and <section> elements play vital roles in structuring web content. <div> elements are used for grouping and organizing sections of content, making it easier to apply CSS styles and JavaScript functionality. <section> elements define thematic sections within a webpage, enhancing its semantic structure and aiding in content organization. In every webpage <div> and <section> are used.

To set up and run the HTML by using VS Code. Create a folder -> open the folder with VS Code. Create a file and name it index.html. Every HTML file has

- <!DOCTYPE html> This declaration specifies that the document is written in HTML5.
- <html lang=”en”></html> The root element that wraps the entire HTML document.
- <head></head> This section contains metadata about the document, including the page title, which is displayed in the browser's tab.
- <body></body> The body of the HTML document contains the visible content of the web page.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My First Web Page</title>
</head>
<body>
    <h1>My name is Sami</h1>
    <p>This is a paragraph of text.</p>
</body>
</html>
```

Figure 10. HTML document.

The complexity of HTML files escalates significantly with the incorporation of extensive content, the application of CSS for intricate styling, and the introduction of JavaScript for heightened interactivity. Nevertheless, the fundamental structure of HTML documents remains constant. By preserving this essential framework, one can store the code within a text file bearing the ".html" extension and subsequently access a web browser to visualize the resultant web page. By adding the above code example lines to the HTML file, will get the header, paragraph, image, form, list of contents, link, and button.

3.1.2 How does HTML work?

HTML works as a structured markup language that defines the content and structure of web pages. Web browsers read and interpret HTML documents to display the content to users, and HTML is a fundamental technology in web development, often combined with CSS for styling and JavaScript for interactivity.

3.2 Introduction to CSS

CSS, or Cascading Style Sheets, is a core web development technique that allows developers to control the visual display and layout of web pages. It is the style language for HTML documents, enabling them to specify how information should appear in a web browser. CSS is crucial in web development as it empowers developers to manage the visual presentation of web content, create attractive and responsive designs, and enhance the overall user experience of websites.

3.2.1 Purpose of CSS

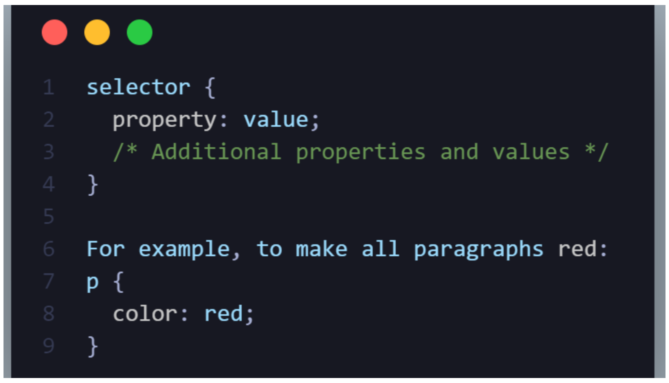
CSS is used to separate the content (HTML) and presentation (styling) of a web page. It allows definition of how elements such as text, images, and other HTML elements should appear on the screen or in print.

3.2.2 How does CSS work?

CSS uses a set of rules and selectors to target specific HTML elements. Selectors identify which HTML elements to style, while rules define how those elements should be styled. CSS rules consist of property-value pairs, where properties specify what to style (e.g. color, font-size), and values specify how to style it (e.g. red, 16px).

CSS Syntax

- A CSS rule typically follows this structure:



```
selector {
  property: value;
  /* Additional properties and values */
}

For example, to make all paragraphs red:
p {
  color: red;
}
```

Figure 11. CSS properties and values.

Cascading Nature:

- CSS is "cascading" meaning that styles can be inherited or overridden.
- Styles can come from different sources, such as external style sheets, internal styles (within HTML), or inline styles (directly within HTML elements).
- Specificity and the order of CSS rules affect which styles take precedence.

Selectors:

- CSS offers various selectors to target specific HTML elements:
  - Type selectors (e.g. p for paragraphs).
  - Class selectors (e.g. .highlight for elements with a specific class).

  - ID selectors (e.g. #header for elements with a specific ID).
  - Attribute selectors (e.g. [type="button"] for elements with a specific attribute value).
  - Combining selectors for more specific targeting.

CSS Properties:

- CSS provides a wide range of properties for styling elements, including text properties (e.g. color, font-size), layout properties (e.g. margin, padding), and (e.g. justify-content, justify-items).

CSS Preprocessors:

- Developers often use CSS preprocessors such as Sass or LESS to enhance CSS with variables, nesting, and functions for more efficient and maintainable styles.

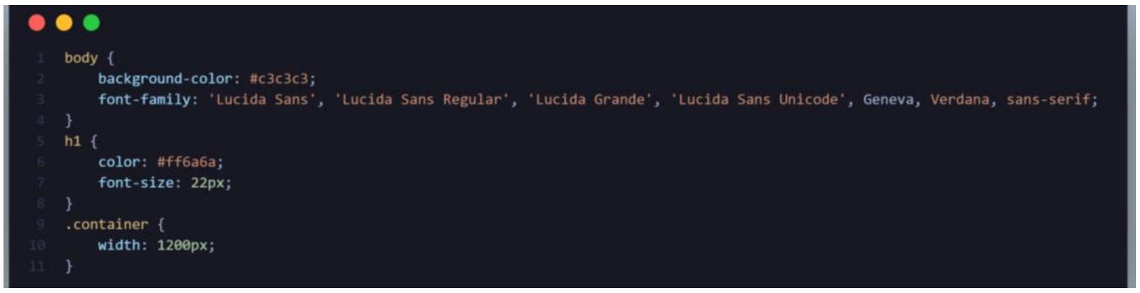
Browser Compatibility:

- CSS is widely supported by modern web browsers, but some features may require vendor-specific prefixes for compatibility.

Responsive Design:

- CSS is crucial for creating responsive web designs that adapt to different screen sizes and devices e.g. mobiles, tablets, desktops, and laptops.

In order to apply styling to a website, the initial step involves the creation of a CSS file, typically named "style.css," within the same directory as the "index.html" file. Within the "index.html" file, it is imperative to insert the following line into the <head> section: <link rel="stylesheet" href="style.css">. Notably, the href attribute within this line should precisely match the name of the CSS file. Crucially, to ensure the successful establishment of this connection, it is vital that both the HTML and CSS files are located within the same folder or directory. Failure to adhere to this placement will result in a failed connection between the HTML and CSS files, and the desired styling will not be applied. (Hwangbo & Lee, 2008)



```css
body {
    background-color: #c3c3c3;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
}
h1 {
    color: #ff6a6a;
    font-size: 22px;
}
.container {
    width: 1200px;
}
```

Figure 12. CSS properties and values.

The stylesheet presented here outlines a set of style rules that dictate the visual presentation of elements in an HTML document. These rules are applied to specific HTML elements using their corresponding selectors, and they are fundamental in enhancing the aesthetics and readability of a web page.

Body Element Styling: The <body> element, which encompasses the entire content of the web page, is configured with several properties. The background-color property specifies a background color, in this case, a shade of gray represented by the hex color code #c3c3c3. This creates an aesthetically pleasing background for the entire page.

In the context of web design, CSS provides a wide array of keywords and properties for achieving desired visual effects on web pages. Memorizing all of them from the outset may seem daunting, but the key to mastering CSS lies in practical experience and gradual learning.

Among the numerous CSS keywords, several play pivotal roles and are commonly employed. These include 'margin' and 'padding' for managing spacing and layout, 'width' and 'height' to control element dimensions, 'text-align' for text alignment, 'font-size' to adjust text size, 'background-color' for background presentation, and 'color' to determine text color.

This selection of fundamental CSS keywords forms a solid foundation for web design. By actively using and applying these keywords in diverse projects, individuals can steadily develop a profound understanding of CSS and its impact on the appearance and structure of web content. With time and practice, these keywords become second nature, empowering web designers to craft visually appealing and well-organized web experiences. (Pan & Ma, 2022)

In the fascinating world of web design and development, Cascading Style Sheets, commonly known as CSS, stand as a cornerstone. CSS empowers web designers to transform the appearance and layout of web content, making it visually appealing and user-friendly A delve into the realm of CSS, allows the discovery of the versatile tools and techniques that facilitate the creation of stunning web experiences.

CSS operates through a system of rules that dictate how HTML elements are presented on the screen. Among these rules, class and ID attributes are fundamental components. Class attributes are used to group and categorize elements with shared characteristics, while ID attributes provide a unique identifier for a specific element. By assigning classes or IDs to elements within an HTML document, these elements can be precisely targeted and styled using CSS.

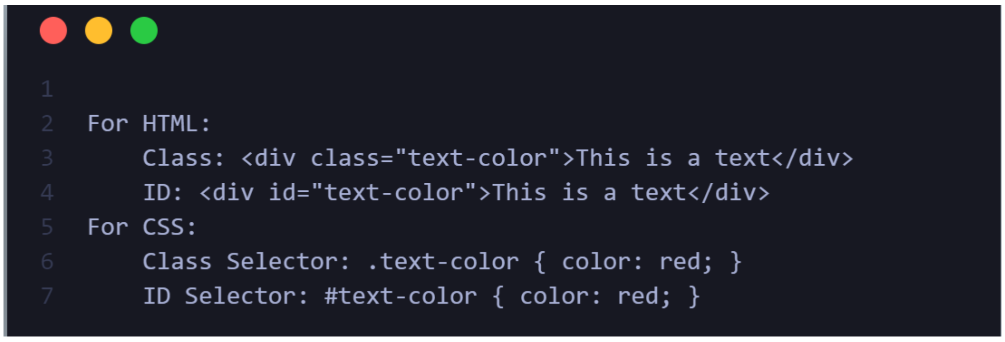
For instance, a 'class' attribute such as class="container" can be applied to multiple elements, allowing a consistent definition of styling for all elements within that class. Conversely, an 'ID' attribute, identified as id="header", creates a unique hook for a single element, enabling the application of distinct styles to that element.

Class attributes, exemplified by class="container", serve as a grouping mechanism that allows multiple elements to share common styling. Applying the same class to various elements within an HTML document ensures uniformity in the presentation of those elements.

On the other hand, ID attributes, denoted by id="header", offer a unique identifier for a specific element. This distinct identifier enables precise targeting and styling of individual elements within a web page.

Class and ID attributes, as integral components of CSS, play a pivotal role in facilitating the creation of aesthetically pleasing and organized web interfaces.

Their use simplifies the process of applying consistent and tailored styles to various elements, contributing to the overall cohesiveness and visual appeal of web content.



```

For HTML:
    Class: <div class="text-color">This is a text</div>
    ID: <div id="text-color">This is a text</div>
For CSS:
    Class Selector: .text-color { color: red; }
    ID Selector: #text-color { color: red; }
```

Figure 13. Created a class in HTML and used it on the CSS.

In CSS, a period (.) is used to select elements by class, and a hash (#) is used to select elements by ID.

4 JavaScript

4.1 Introduction to JavaScript

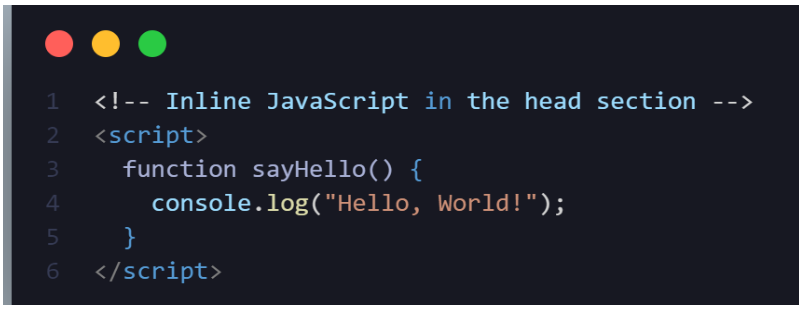
JavaScript is deemed a powerful and widely used programming language, as indicated by the Stack Overflow Developers Survey in 2022. With JavaScript, can construct robust websites, web applications, and games. Moreover, the introduction of Node.js in 2009 has extended JavaScript's capabilities to the server-side, allowing for the development of full-stack applications. This release has empowered developers to build full-stack applications with JavaScript.

The history of JavaScript started in 1995 with the name MOCHA. It underwent subsequent name changes, first to LiveScript and eventually settling on JavaScript. Before JavaScript was introduced, developers struggled to build dynamic websites, using only HTML, with CSS to have structure and a bit of styling, but when JavaScript was released to add programming in the browser, then it started being possible to build powerful dynamic websites with different functionality.

Other major graphical web browsers have since adopted this language, enabling the creation of modern web applications that allow direct interaction without the need for page reloading after each action. JavaScript is also utilized in more traditional websites to enhance interactivity and clever features. It's worth highlighting that JavaScript is unrelated to the programming language called Java. The resemblance in names was driven by marketing decisions rather than sound judgment. During the introduction of JavaScript, the Java language was gaining significant popularity, and someone saw it as an opportunity to associate the two. Consequently, we are now left with the name, even though the two languages are distinct.

4.1.1 How does JavaScript work?

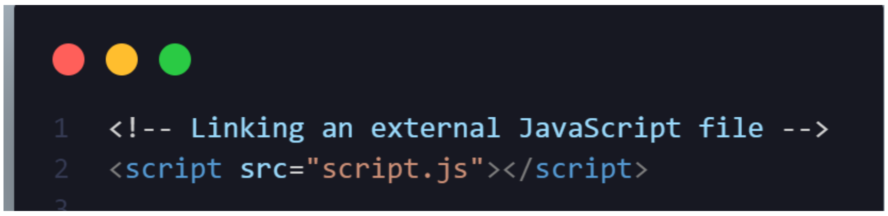
To apply JavaScript to a website, there are two primary methods. The first method is to include JavaScript code directly within the HTML file using inline <script> tags. The <script> tags can be placed either in the <head> section or at the bottom of the HTML file, just before the closing </body> tag. Here's an example of inline JavaScript:



```
<!-- Inline JavaScript in the head section -->
<script>
  function sayHello() {
    console.log("Hello, World!");
  }
</script>
```

Figure 14. Inline JavaScript in the head section.

The second method is to create a separate JavaScript file, typically named "script.js," and link to it in HTML using the <script> tag. This approach promotes clean and organized code by separating JavaScript logic from the HTML content. (Xiao, 2016). Here's how to link an external JavaScript file:



```
<!-- Linking an external JavaScript file -->
<script src="script.js"></script>
```

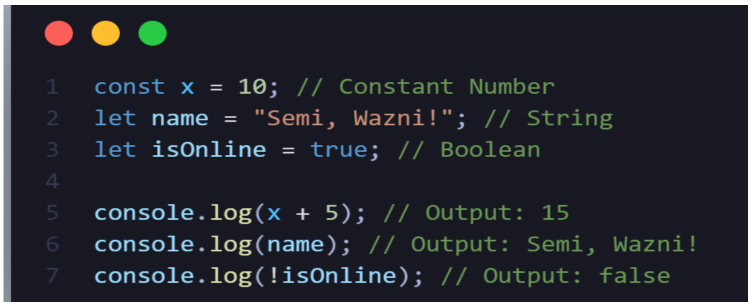
Figure 15. JavaScript “script tag” to connect JavaScript file.

By following this approach, maintains stricter code organization and encapsulates all JavaScript functionality within its own file. It's essential to ensure that the filename in the src attribute matches the name of the JavaScript file exactly. Equally important is that both the HTML and JavaScript files are located in the same directory. Failing to adhere to these guidelines will result in a failed connection between the HTML and JavaScript files, preventing the JavaScript functionality from working as intended.

4.1.2 Variables

Variables serve as containers within a program, associating a name (identifier) with reusable data. They play a vital role in storing and managing data, allowing developers to manipulate, process, and interact with information effectively. The correct usage of variables in JavaScript significantly influences the quality and efficiency of code, making them a central component of the language and an essential topic for in-depth exploration in this research.

- var is a keyword used in pre-ES6 versions of JavaScript.
- let is the preferred way to declare a variable when it can be reassigned.
- const is the preferred way to declare a variable with a constant value, which means the value cannot be changed or replaced.



```javascript
const x = 10; // Constant Number
let name = "Semi, Wazni!"; // String
let isOnline = true; // Boolean

console.log(x + 5); // Output: 15
console.log(name); // Output: Semi, Wazni!
console.log(!isOnline); // Output: false
```

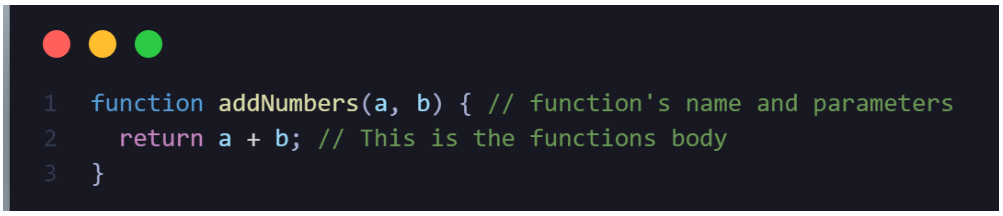
Figure 16. JavaScript variabels.

Uninitialized variables in JavaScript store the primitive data type "undefined." Employing mathematical assignment operators simplifies the computation of new values and facilitates their assignment to the same variable. The + operator serves to concatenate strings, incorporating string values stored in variables. In ECMAScript 6 (ES6), template literals utilize backticks () and ${} to seamlessly interpolate values into a string. The typeof` keyword, when employed, yields the data type of a value, returned as a string.

4.1.3 Functions

Functions serve as a reusable block of code, uniting a sequence of statements to execute a particular task. Within a function's block, a parameter, identified as a named variable, awaits the assignment of the value passed in as an argument when the function is called. This synergy between functions and parameters forms the foundation of effective code organization and execution. (Xiao, 2016).

Declaration of a function:



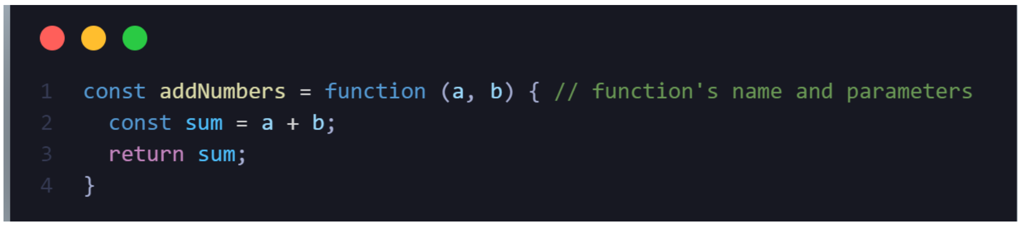
```javascript
function addNumbers(a, b) { // function's name and parameters
  return a + b; // This is the functions body
}
```

Figure 17. JavaScript function.

ES6 introduces innovative approaches to handle arbitrary parameters by utilizing default parameters, enabling the assignment of default values to parameters when no arguments are provided to the function.

Returning a value from a function is accomplished using a return statement, offering a way to convey a specific result or outcome.

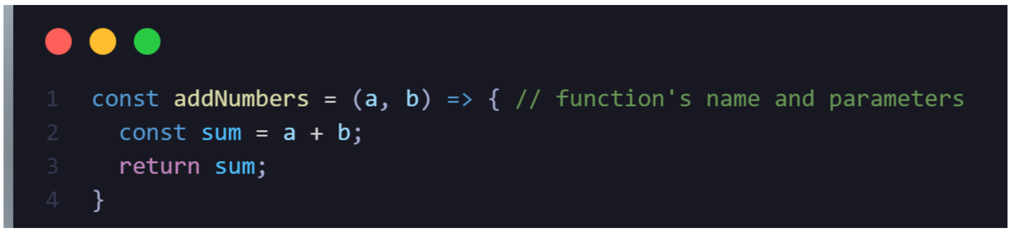
To a function using function expressions:



```javascript
const addNumbers = function (a, b) { // function's name and parameters
  const sum = a + b;
  return sum;
}
```

Figure 18. JavaScript function expressions.

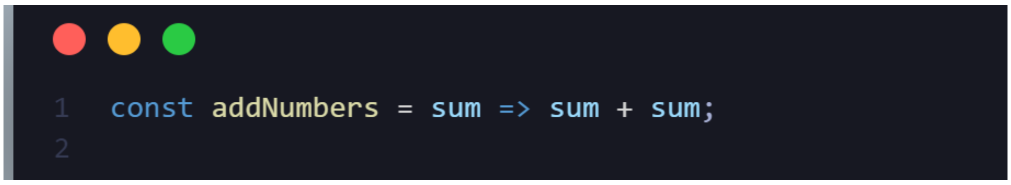
To declare a function using arrow function notation:



```javascript
const addNumbers = (a, b) => { // function's name and parameters
  const sum = a + b;
  return sum;
}
```

Figure 19. JavaScript arrow function.

Function definitions can be streamlined using concise arrow notation:



```javascript
const addNumbers = sum => sum + sum;
```

Figure 20. JavaScript concise arrow function.

Acknowledging the disparities among function expressions, arrow functions, and function declarations is essential. Engaging in extensive JavaScript programming reveals a diverse array of applications for these distinct function types.

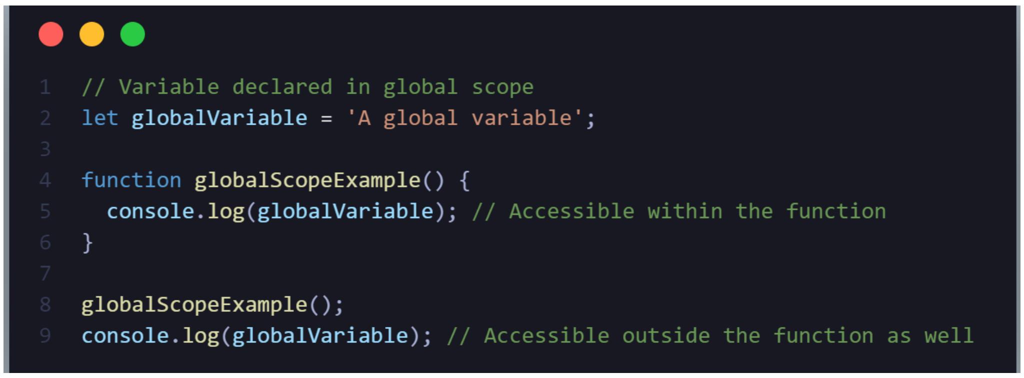
4.1.4 Scope

Scope in JavaScript is a fundamental concept that governs the accessibility of variables, dictated by their declaration within the code. This encompasses the global scope, where variables are universally accessible throughout the entire program, and the block scope, where their accessibility is confined to the specific block in which they are defined.

In the context of programming, scope pollution emerges when an excess of variables occupies a namespace, or when variable names are repetitively used, potentially leading to confusion and errors. A judicious management of variable scoping, ensuring tight control, is pivotal for sustaining a codebase characterized by cleanliness, organization, and modularity.

Understanding and implementing effective scoping practices not only contributes to code readability and maintainability but also fosters the development of robust and comprehensible JavaScript programs.

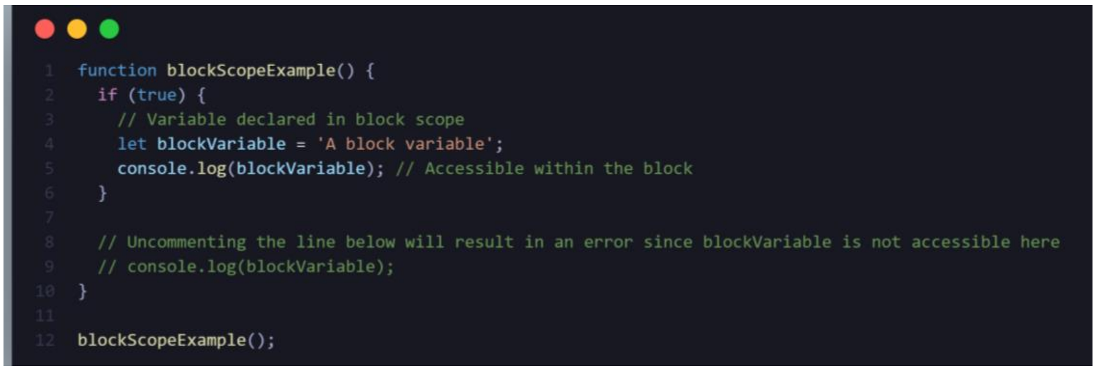
4.1.4.1 Global Scope



```javascript
// Variable declared in global scope
let globalVariable = 'A global variable';

function globalScopeExample() {
  console.log(globalVariable); // Accessible within the function
}

globalScopeExample();
console.log(globalVariable); // Accessible outside the function as well
```

Figure 21. JavaScript Scope.

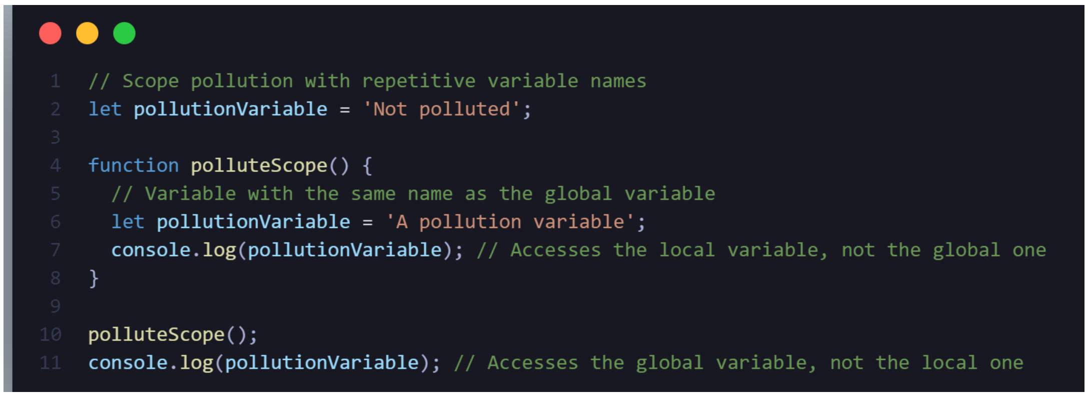
4.1.4.2 Block Scope



```javascript
function blockScopeExample() {
  if (true) {
    // Variable declared in block scope
    let blockVariable = 'A block variable';
    console.log(blockVariable); // Accessible within the block
  }

  // Uncommenting the line below will result in an error since blockVariable is not accessible here
  // console.log(blockVariable);
}

blockScopeExample();
```

Figure 22. JavaScript Block Scope.

4.1.4.3 Scope Pollution



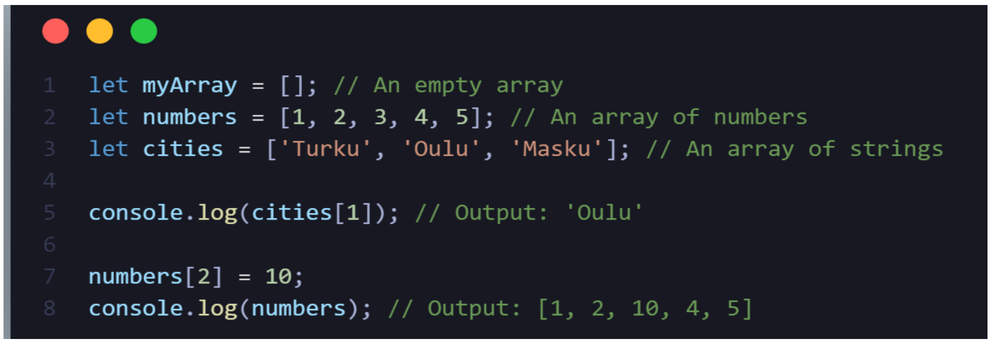
```javascript
// Scope pollution with repetitive variable names
let pollutionVariable = 'Not polluted';

function polluteScope() {
  // Variable with the same name as the global variable
  let pollutionVariable = 'A pollution variable';
  console.log(pollutionVariable); // Accesses the local variable, not the global one
}

polluteScope();
console.log(pollutionVariable); // Accesses the global variable, not the local one
```

Figure 23. JavaScript Scope Pollution.

4.1.5 Arrays

Arrays in JavaScript serve as ordered lists for data storage. The creation of arrays involves the use of brackets ([]), and each item within an array is assigned a numerical position, or index, commencing from 0. Accessing a specific item within an array is achieved by referencing its index, denoted as myArray[0].

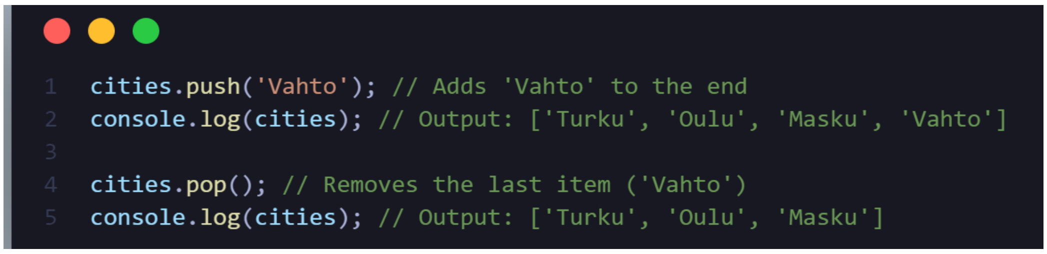
Modifying array elements is also accomplished through their respective indices, such as myArray[0] = 'new string'. The length property of arrays provides information about the number of items contained within.



```javascript
let myArray = []; // An empty array
let numbers = [1, 2, 3, 4, 5]; // An array of numbers
let cities = ['Turku', 'Oulu', 'Masku']; // An array of strings

console.log(cities[1]); // Output: 'Oulu'

numbers[2] = 10;
console.log(numbers); // Output: [1, 2, 10, 4, 5]
```

Figure 24. Defined an Array.

Arrays are equipped with methods such as .push() and .pop() for adding and removing items, respectively. Additionally, arrays feature an array of methods such as .slice() and .shift(), each serving specific purposes. Comprehensive documentation is available on the Mozilla Developer Network.



```javascript
cities.push('Vahto'); // Adds 'Vahto' to the end
console.log(cities); // Output: ['Turku', 'Oulu', 'Masku', 'Vahto']

cities.pop(); // Removes the last item ('Vahto')
console.log(cities); // Output: ['Turku', 'Oulu', 'Masku']
```

Figure 25. Array methods.

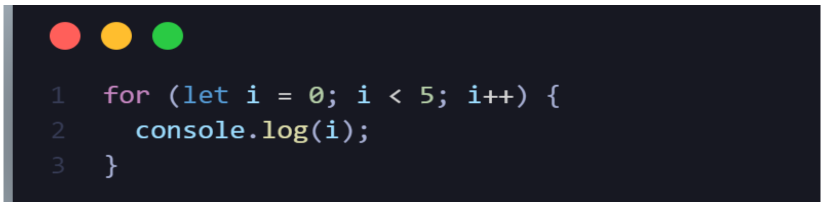
It is pertinent to note that certain array methods are mutating, thereby altering the array, while others are non-mutating. Clarification on the nature of each method can be sought in the documentation.

Variables containing arrays can be declared using let or const, and although arrays declared with const remain mutable, the variable itself cannot be reassigned. Importantly, arrays mutated within a function retain those changes outside the function scope. Arrays also offer the possibility of nesting, allowing arrays to be embedded within other arrays. Accessing elements within nested arrays involves chaining indices through bracket notation.

4.1.6 Loops

Loops execute repetitive tasks, alleviating the need for manual coding of such processes on each occurrence. This encompasses the creation of for loops utilizing an iterator variable for incremental or decremental operations.

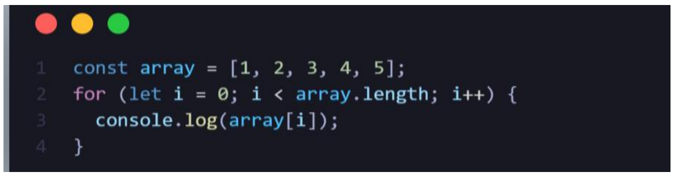
For Loops with Iterator Variable:



```javascript
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

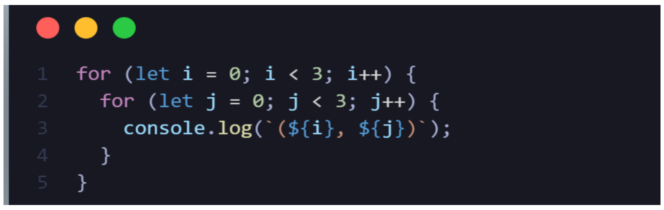
Figure 26. JavaScript for loop.

Additionally, the application of for loops to iterate through arrays is explored. Delving further, nested for loops are examined, representing loops encapsulated within one another. While loops provide flexibility in defining various stopping conditions, emphasizing their critical role in averting infinite loops. Furthermore, the do...while loop structure ensures the execution of code at least once, with the stopping condition assessed only after the initial iteration. Finally, the break keyword is introduced as a mechanism for prematurely exiting a loop during its block execution.



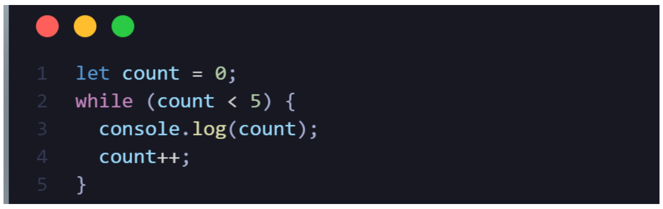
```
const array = [1, 2, 3, 4, 5];
for (let i = 0; i < array.length; i++) {
  console.log(array[i]);
}
```

Figure 27. For Loop to Iterate Through an Array.



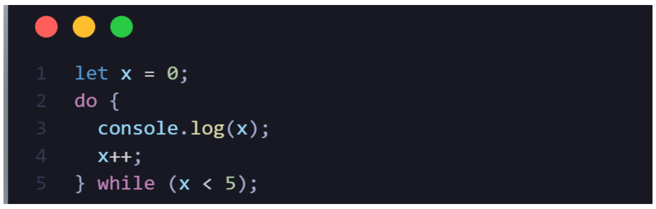
```
for (let i = 0; i < 3; i++) {
  for (let j = 0; j < 3; j++) {
    console.log(`(${i}, ${j})`);
  }
}
```

Figure 28. Nested For Loops.



```
let count = 0;
while (count < 5) {
  console.log(count);
  count++;
}
```

Figure 29. While Loops with Stopping Conditions.



```
let x = 0;
do {
  console.log(x);
  x++;
} while (x < 5);
```

Figure 30. do...while Loop.

5 ReactJS

5.1 Introduction to React

React is a JavaScript library which plays a pivotal role in the development of modern web application interfaces. It offers developers a comprehensive toolkit to create responsive, dynamic, and user-friendly web applications. This library streamlines the web development process, enhancing its structure and efficiency.

The core concept of React revolves around the use of components. Components are not isolated entities; instead, they are reusable elements that can be seamlessly integrated across the entire application. This component-based architecture promotes reusability and maintainability, making it easier to manage complex web applications.

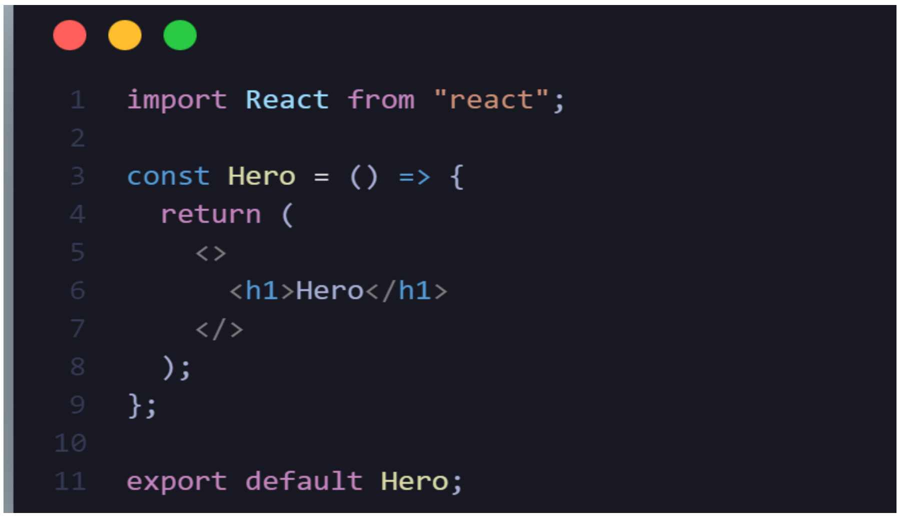
React's versatility extends to the development of full-stack applications, making a compelling choice for businesses across diverse industries. It caters to a wide range of needs, from interactive user interfaces to robust back-end functionality. (Roldán, 2023).

In recent years, ReactJS has witnessed widespread adoption, with numerous major corporations selecting it as their technology of choice. Companies such as Netflix, Facebook, Instagram, Airbnb, Twitter, and PayPal have harnessed the power of React to build robust and scalable applications. This adoption is a testament to ReactJS's ability to meet the demands of today's dynamic and competitive digital landscape.

As React continues to evolve and the community around it thrives, it remains a leading choice for developers and businesses looking to stay at the forefront of web application development.

5.1.1 Components

In React, components play a pivotal role, enabling the division of a complex application into smaller and manageable parts. Each component is given a name and resides in the "src" folder. Component names conventionally begin with a capital letter, for example, "Home.js." Components generally adhere to a similar structure, as depicted in the following Figure 31:



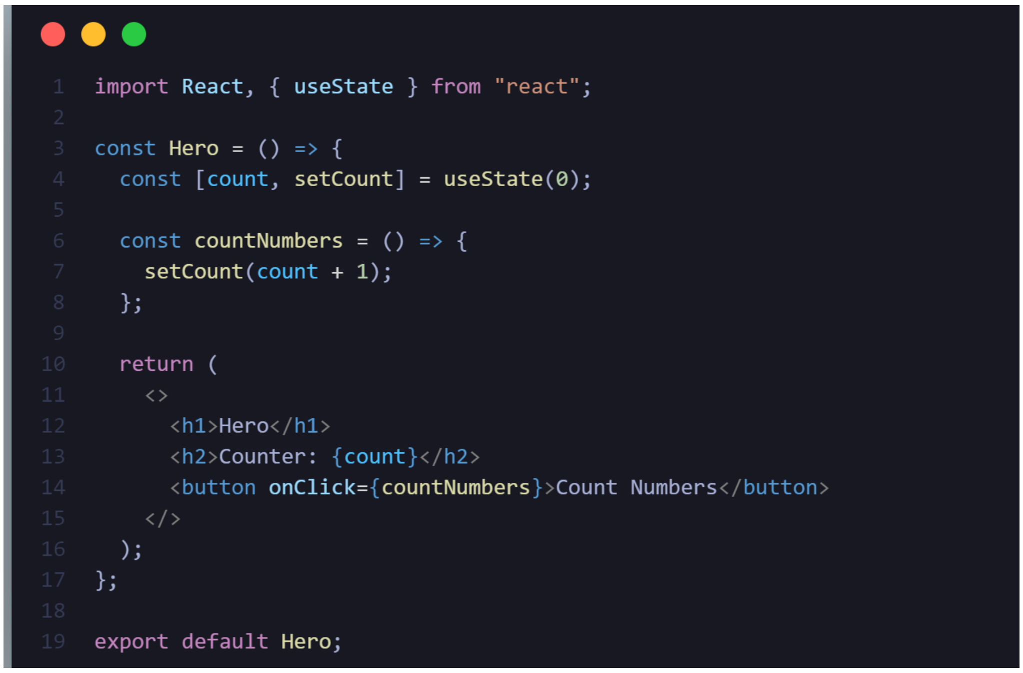
```
import React from "react";

const Hero = () => {
  return (
    <>
      <h1>Hero</h1>
    </>
  );
};

export default Hero;
```

Figure 31. React Component.

This structure is characteristic of React components, making it easier to create and manage various parts of the application.

5.1.2 Hooks

In React, hooks provide a powerful mechanism for managing state and side effects within functional components. They enable developers to access features such as state, lifecycle methods, and context within functional components, thereby enhancing their flexibility and reusability. Hooks are included as functions like useState or useEffect and can be used to augment the functionality of functional components without the need for class components. The use of hooks has become a standard practice in modern React development, allowing for more concise and organized code.



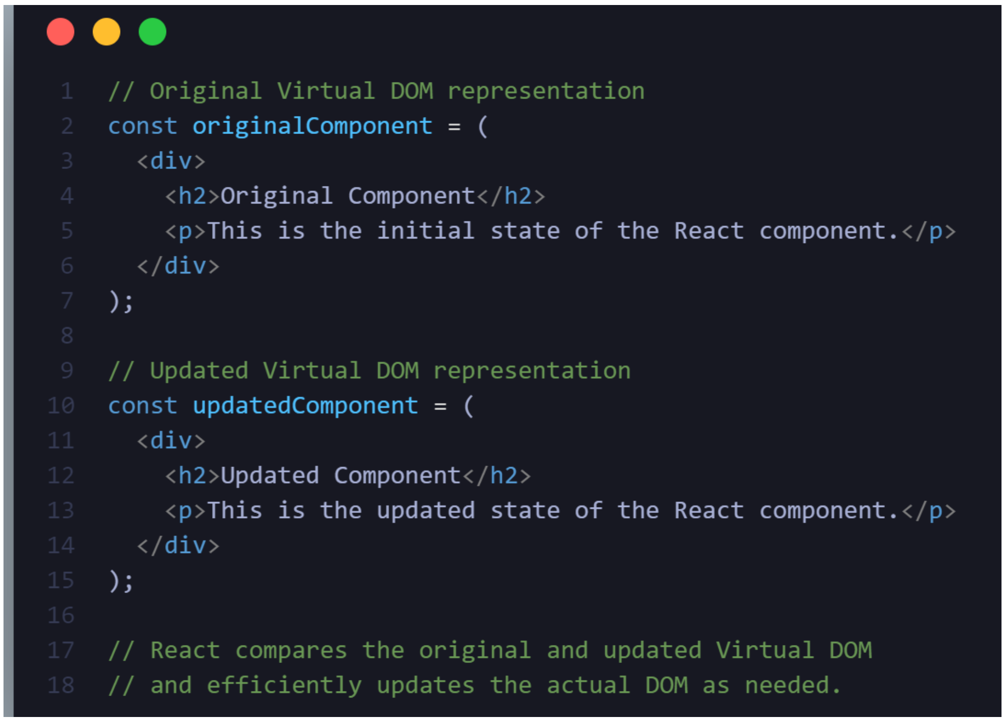
```
import React, { useState } from "react";

const Hero = () => {
  const [count, setCount] = useState(0);

  const countNumbers = () => {
    setCount(count + 1);
  };

  return (
    <>
      <h1>Hero</h1>
      <h2>Counter: {count}</h2>
      <button onClick={countNumbers}>Count Numbers</button>
    </>
  );
};

export default Hero;
```

Figure 32. React Component with a useState Hook.

Hooks have become an integral part of React development, empowering developers to create more efficient and maintainable components.

5.1.3 Virtual DOM

React uses a method known as the Virtual DOM, an in-memory representation mirroring the real DOM. It undertakes a comparison between the existing and new tree structures to minimize the necessity for actual DOM updates. This optimization process, termed reconciliation, is harnessed by both React DOM and React Native in crafting user interfaces for their respective platforms. (Roldán, 2023).



```
// Original Virtual DOM representation
const originalComponent = (
  <div>
    <h2>Original Component</h2>
    <p>This is the initial state of the React component.</p>
  </div>
);

// Updated Virtual DOM representation
const updatedComponent = (
  <div>
    <h2>Updated Component</h2>
    <p>This is the updated state of the React component.</p>
  </div>
);

// React compares the original and updated Virtual DOM
// and efficiently updates the actual DOM as needed.
```

Figure 33. React DOM.

5.1.4 Declarative Syntax

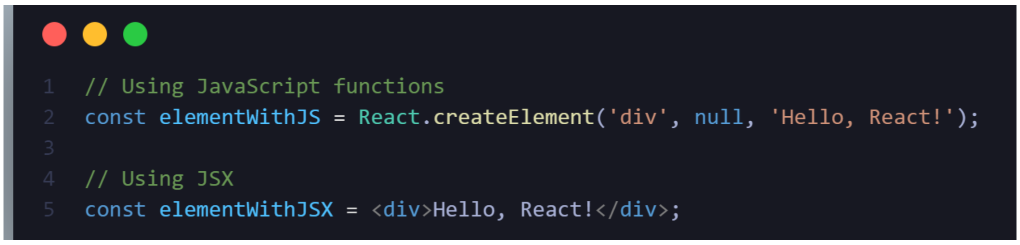
In React, the declarative syntax refers to the approach of describing what the UI should look like and ReactJS takes care of updating the UI to match that description. It contrasts with the imperative approach, where the focus is on detailing the step-by-step process of achieving a result.

With declarative syntax, developers express the desired state or appearance of the user interface and ReactJS efficiently manages the underlying updates to achieve that state. This simplifies the development process, making code more intuitive and readable, and allows React to optimize performance by handling the necessary updates behind the scenes. Overall, the declarative syntax in React contributes to a more straightforward and maintainable way of building user interfaces.

5.1.5 Using JSX

React offers two approaches for defining elements: utilizing JavaScript functions and employing JSX, an optional XML-like syntax.

The utilization of JSX is a common stumbling block. The integration of JavaScript and HTML within examples on the homepage can initially appear unfamiliar to many. (Roldán, 2023).



```
// Using JavaScript functions
const elementWithJS = React.createElement('div', null, 'Hello, React!');

// Using JSX
const elementWithJSX = <div>Hello, React!</div>;
```

Figure 34. React JSX.

5.2 Getting Started with React. Build a Portfolio with React.

The project comprises a portfolio divided into six components for a more manageable structure. These components include Hero, Skills, About, Project, Nav, and Contact, each housing pertinent information. The application leverages HTML, CSS, and the React library to facilitate the creation of a dynamic web application. Notably, the Contact component incorporates EmailJS functionality, enabling the application to send emails seamlessly through a form submission process.

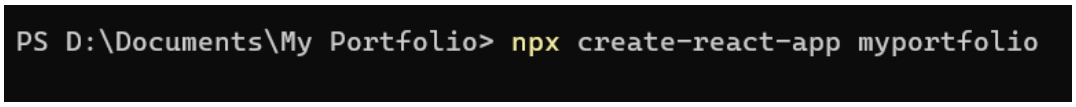
To set up the project locally, create a folder on the computer named "My Portfolio" and access this folder using the command line (cmd/PowerShell). Proceed with the following steps to initiate the React project within this directory:

Commencing a React project necessitates the establishment of the developmental environment, the creation of the project structure, and the articulation of research objectives and scope. The outlined procedure encompasses setting up the development environment, creating the React project, defining objectives, structuring the project, and managing data and state. The entire process entails documentation, version control, and a continual review, culminating in the presentation and defence of the project. This comprehensive approach ensures a systematic and well-documented progression for the successful completion of a React-based project. (Sengupta et al., 2016).

- Setting Up the Development Environment:
  The foundational prerequisite is to ensure the readiness of the development environment, encompassing the installation of Node.js and npm. These essential components can be installed via the official platform. Node.js and npm(Node Package Manager). React applications are typically built and managed with Node.js and npm. Download and install Node.js, which includes npm, from the official website: nodejs.org.
- Initiating a React Project:

  To initiate a React project, the "Create React App" tool is recommended. Execution of the ensuing command in the terminal is recommended:

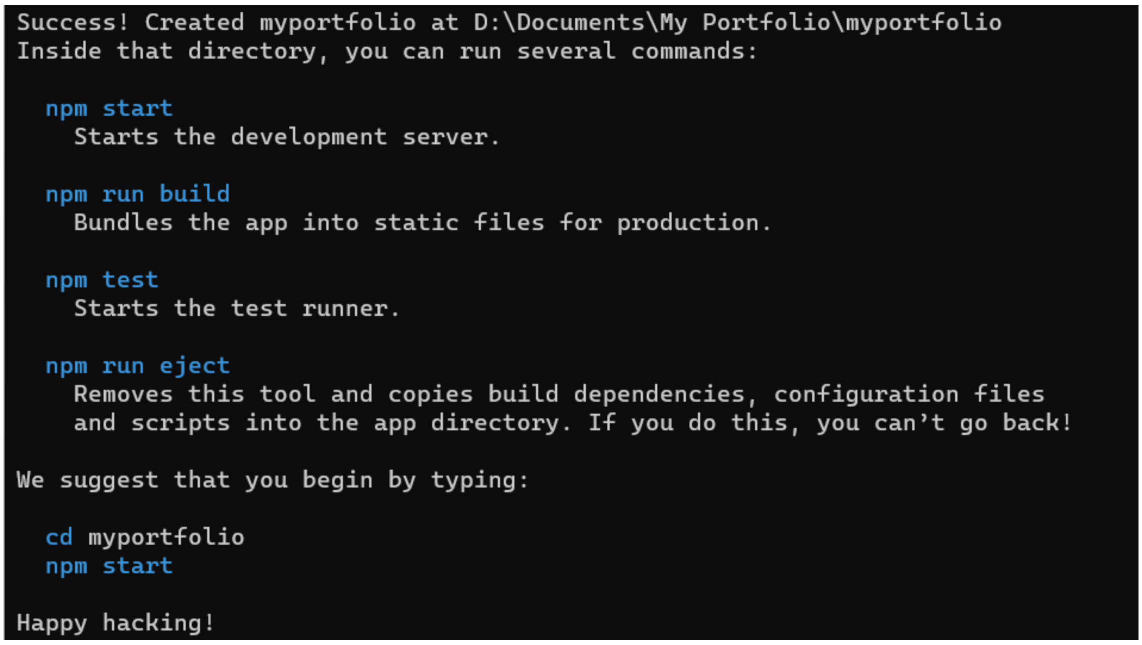
  Create a folder in the computer then open the Command Prompt(cmd)



```
PS D:\Documents\My Portfolio> npx create-react-app myportfolio
```

Picture 1. React installation.

Customization of the project name is accommodated. This command effectuates the establishment of a rudimentary React project replete with a predefined directory hierarchy, dependencies, and an operational development server.



```
Success! Created myportfolio at D:\Documents\My Portfolio\myportfolio
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd myportfolio
  npm start

Happy hacking!
```

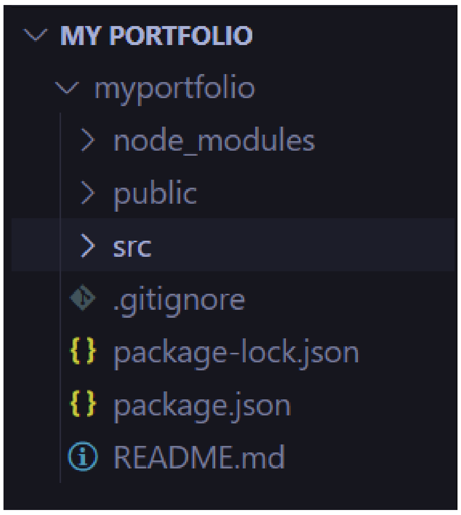
Picture 2. React installation done.

To commence the project, execute the "npm start" command. This command initiates the project, making it accessible via localhost:3000. An essential prerequisite for this process is the utilization of a text editor, which is necessary for the creation and modification of code.

- Text Editor like VS code or similar. In this thesis, VS code will be used. Download and install the VS code from the official website: visualstudio.com/download.
- Project Structural Framework:
  The extant project structure, as furnished by "Create React App," warrants perusal. Familiarity with the principal directories is indispensable, with a focus on "src" for source code and "public" for publicly accessible assets such as HTML and images. The incorporation of supplementary directories, aligned with the specific project requirements, is a plausible option.

  In the project directory, two main files are present: 'public/index.html' and 'src/App.js.' Components should be created and imported into 'App.js.' The

React environment will automatically generate the HTML markup in the 'index.html' file. The 'index.html' file functions as the container for all project code, subsequently rendering it in the browser.



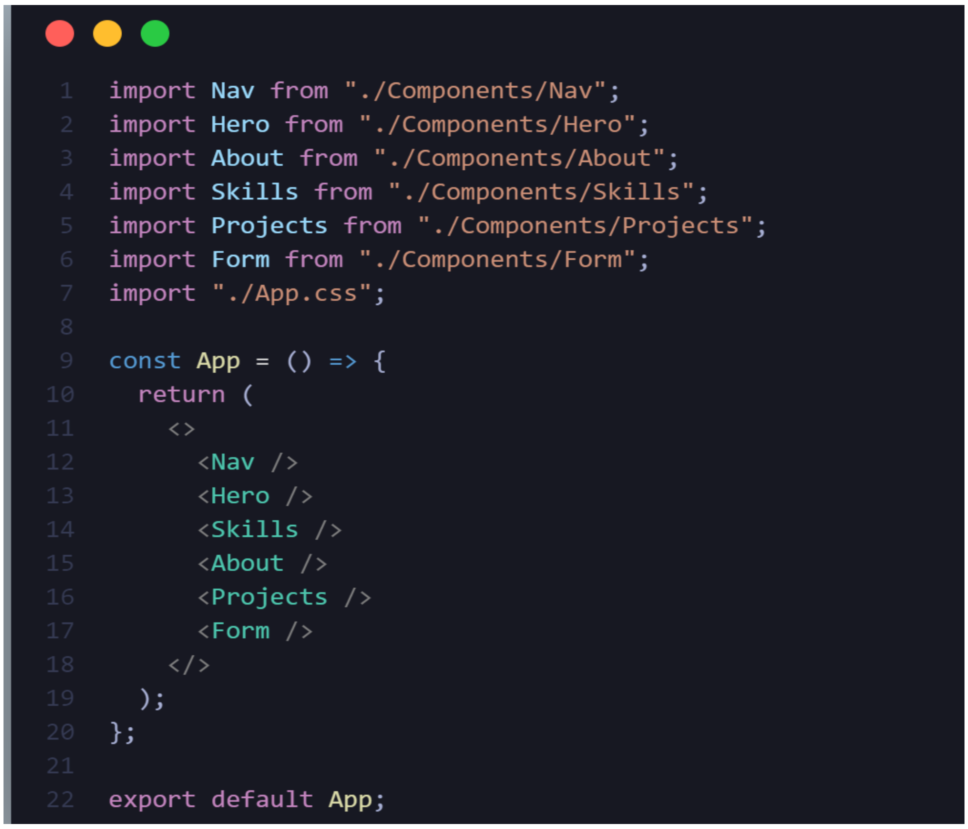
```
MY PORTFOLIO
  myportfolio
    node_modules
    public
    src
    .gitignore
    package-lock.json
    package.json
    README.md
```

Picture 3. React default folders.

- Project Code Development:
  The inception of the React application code is to be undertaken, with the principal sphere of activity typically being the "src" directory. The development of components, state management, and the creation of requisite views is imperative. Adherence to established best practices and the maintenance of code organization to facilitate debugging and future maintenance are paramount.

5.2.1 Use of Component and Nav

In the "src" folder, a Components folder with six components was created to handle the Hero.js, Skills.js, About.js, Projects.js, Nav.js, and Contact.js. Once the components were created, they were set up in "App.js" to display them within the project.

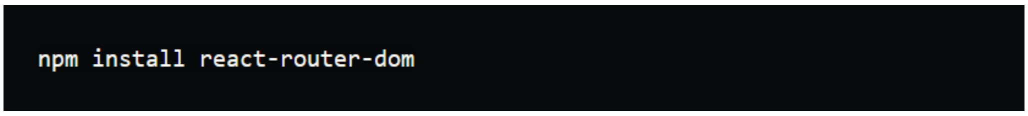


```
import Nav from "./Components/Nav";
import Hero from "./Components/Hero";
import About from "./Components/About";
import Skills from "./Components/Skills";
import Projects from "./Components/Projects";
import Form from "./Components/Form";
import "./App.css";

const App = () => {
  return (
    <>
      <Nav />
      <Hero />
      <Skills />
      <About />
      <Projects />
      <Form />
    </>
  );
};

export default App;
```

Figure 35. App.js file to display the components within the project.

Each component has a similar setup, following the default structure in React. The initial lines of each component should include necessary imports, such as `import React from "react";`. It is essential to import all components before returning them to the application. If the components are not imported, the application will generate an error indicating that the module is not defined.

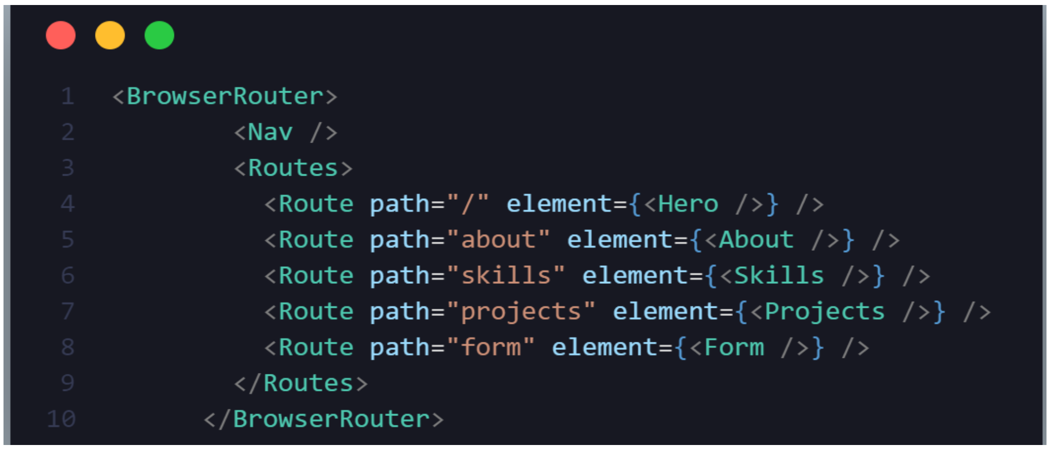
To ensure a seamless experience and prevent frequent loading when navigating between components, the project utilizes Router v6. This choice facilitates smooth transitions between components without reloading pages each time. Setting up the Router is necessary for its proper functionality. First, the react-router dom needs to be set by running in the terminal:



```
npm install react-router-dom
```

Picture 4. Install react router dom

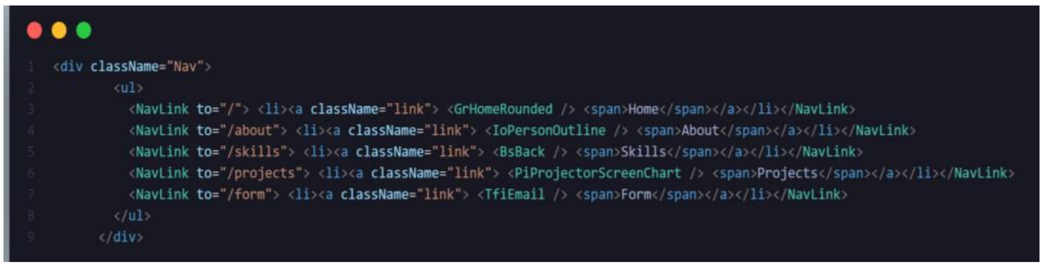
This command installs the react-router-dom package, which is a library providing a set of navigational components for React applications. It enables the implementation of routing and navigation in a declarative manner.



```
<BrowserRouter>
        <Nav />
        <Routes>
          <Route path="/" element={<Hero />} />
          <Route path="about" element={<About />} />
          <Route path="skills" element={<Skills />} />
          <Route path="projects" element={<Projects />} />
          <Route path="form" element={<Form />} />
        </Routes>
      </BrowserRouter>
```

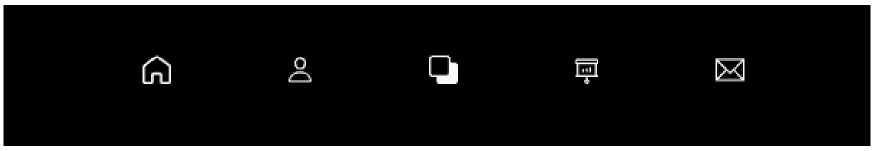
Figure 36. React Router applied in App.js.

To ensure the navigation functionality, a Nav component is created, serving as a container for all components and enabling them to be clickable for seamless navigation between different sections. The implementation involves using a Router and dividing the project into several components. When a specific component name is clicked, the corresponding component is displayed, ensuring efficient navigation throughout the project.



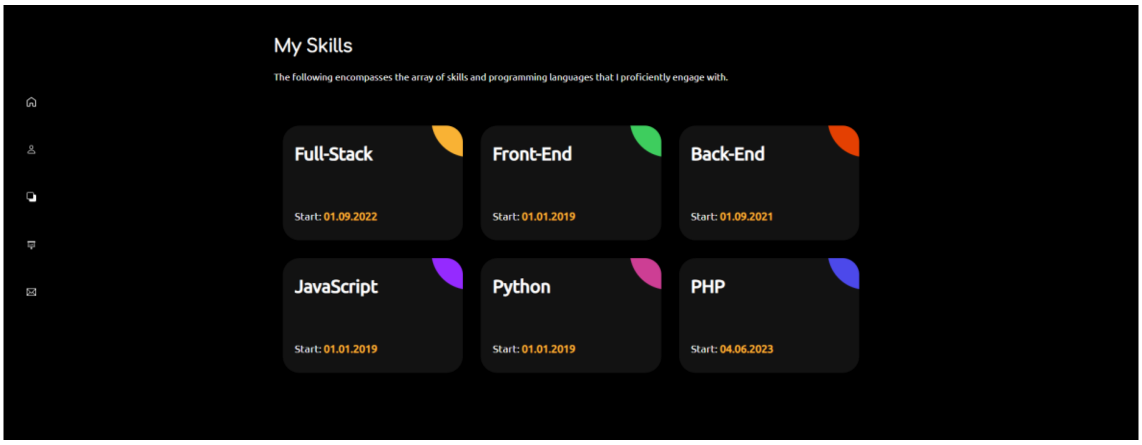
```
<div className="Nav">
        <ul>
          <NavLink to="/"> <li><a className="link"> <GrHomeRounded /> <span>Home</span></a></li></NavLink>
          <NavLink to="/about"> <li><a className="link"> <IoPersonOutline /> <span>About</span></a></li></NavLink>
          <NavLink to="/skills"> <li><a className="link"> <BsBack /> <span>Skills</span></a></li></NavLink>
          <NavLink to="/projects"> <li><a className="link"> <PiProjectorScreenChart /> <span>Projects</span></a></li></NavLink>
          <NavLink to="/form"> <li><a className="link"> <TfiEmail /> <span>Form</span></a></li></NavLink>
        </ul>
      </div>
```

Figure 37. React Application navbar for all components.



Picture 5. MyPortfolio Application's Navbar.

The project's components are structured with distinct sections of content, defined using div elements for customizable styling. In Picture 6 below, the "Skills" component is showcased, emphasizing the organization of its content.



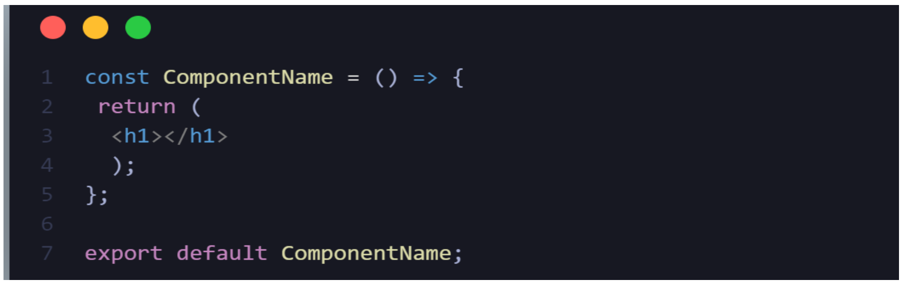
My Skills

The following encompasses the array of skills and programming languages that I proficiently engage with.

Full-Stack Start: 01.09.2022	Front-End Start: 01.01.2019	Back-End Start: 01.09.2021
JavaScript Start: 01.01.2019	Python Start: 01.01.2019	PHP Start: 04.06.2023

Picture 6. Skills.js Component.

Furthermore, every component within the application adheres to a standardized React structure, exemplified by the generic template:



```
const ComponentName = () => {
 return (
  <h1></h1>
  );
};

export default ComponentName;
```

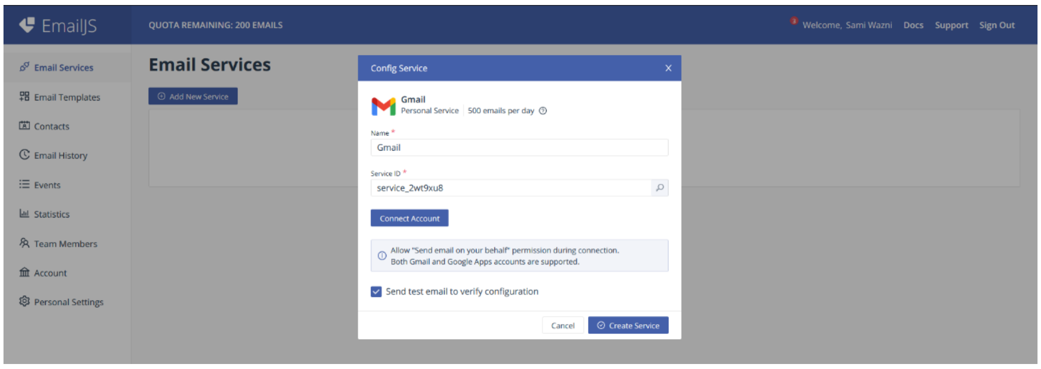
Figure 38. React Component.

Each of these components encompasses distinct functionalities tailored to fulfil its specific role within the application, contributing to a modular and maintainable codebase.

5.2.2 Use of EmailJS

In this Application, as there is no server (Back-end), an email tool was employed. EmailJS serves as a backend-as-a-service, enabling users to send emails directly from the client side without the need to set up a server.

The process involves creating an account and configuring Gmail as an email server. This includes establishing a server connection linked to the designated email address responsible for receiving the emails, Picture 7



Gmail
Personal Service | 500 emails per day

Name *
Gmail

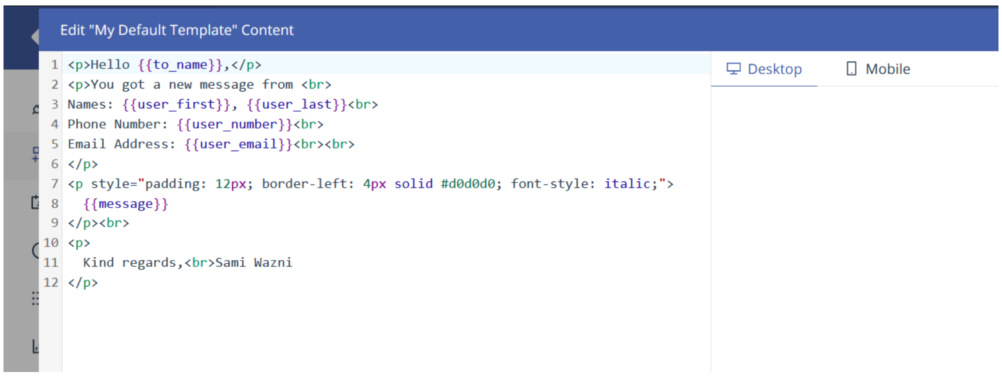
Service ID *
service_2wt9xu8

Allow "Send email on your behalf" permission during connection.
Both Gmail and Google Apps accounts are supported.

☑ Send test email to verify configuration

Picture 7. Set up an Email server.

Following the connection, an email template was crafted to specify the desired format for receiving emails.



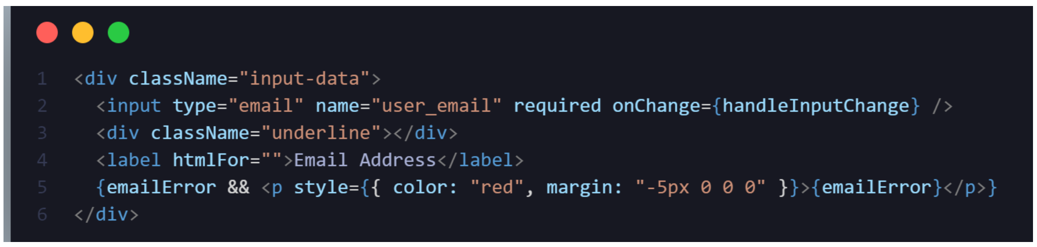
```
<p>Hello {{to_name}},</p>
<p>You got a new message from <br>
Names: {{user_first}}, {{user_last}}<br>
Phone Number: {{user_number}}<br>
Email Address: {{user_email}}<br><br>
</p>
<p style="padding: 12px; border-left: 4px solid #d0d0d0; font-style: italic;">
  {{message}}
</p><br>
<p>
  Kind regards,<br>Sami Wazni
</p>
```

Picture 8. Set up the email.

Following the account and template configuration, the required React packages for email reception were installed using the npm package manager: `npm install @emailjs/browser`.

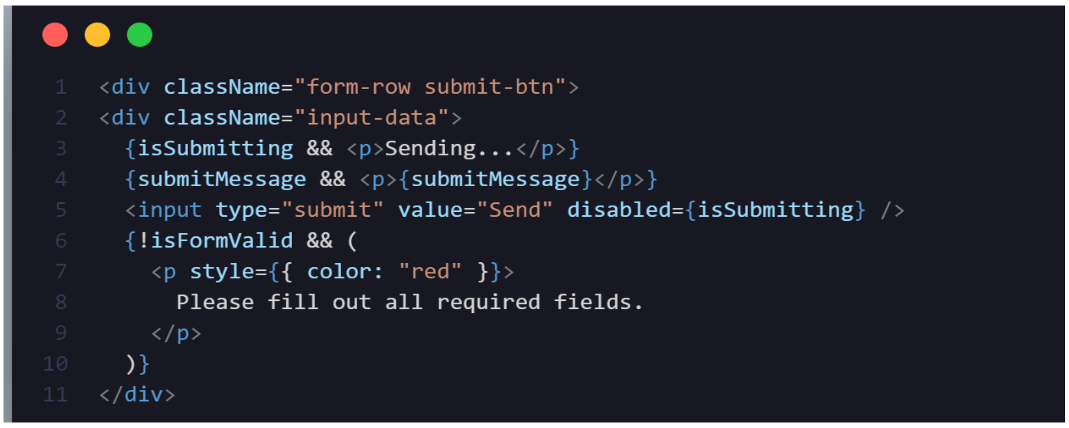
In the form inputs, specific attributes, as exemplified in Picture 8 of the email template setup, were employed. For instance, the attribute name="user_first" ensures that the first name input captures its value, is subsequently sent to the EmailJS server, and forwarded to the designated email address. Each input field is assigned its unique attribute to handle diverse values, Figure 39.

The default form initially features only name and email fields. However, for enhanced functionality, new fields were systematically introduced. Rigorous validation checks were implemented to ensure the accuracy and completeness of user inputs. Furthermore, a progress indicator was seamlessly incorporated, enriching the user experience by providing real-time feedback during the email transmission process.



```
<div className="input-data">
  <input type="email" name="user_email" required onChange={handleInputChange} />
  <div className="underline"></div>
  <label htmlFor="">Email Address</label>
  {emailError && <p style={{ color: "red", margin: "-5px 0 0 0" }}>{emailError}</p>}
</div>
```

Figure 39. Form for the Application with EmailJS attribute.



```
<div className="form-row submit-btn">
<div className="input-data">
  {isSubmitting && <p>Sending...</p>}
  {submitMessage && <p>{submitMessage}</p>}
  <input type="submit" value="Send" disabled={isSubmitting} />
  {!isFormValid && (
    <p style={{ color: "red" }}>
      Please fill out all required fields.
    </p>
  )}
</div>
```

Figure 40. Email transmission process.

Upon submission of the form, an email notification will be generated, containing the specified attributes. The email format aligns with the predefined template as depicted in Picture 8. The email will be structured as follows:

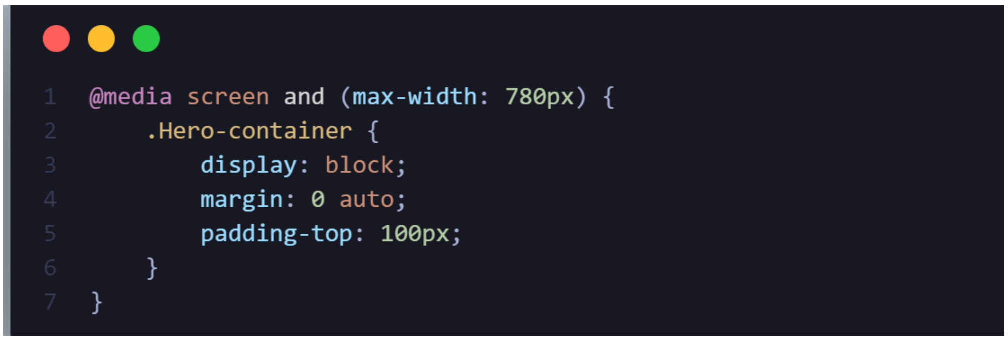
```
Hello,

You have received a new message from:

Names: Sami, Wazni

Phone Number: 0400000000

Email Address: sami@edu.turkuamk.fi

Message: This is an email for the test of the thesis matter.
```

5.2.3 Use of CSS

This application employs pure CSS for the purpose of attaining customized styling tailored to specific requirements. The CSS file is systematically organized with distinct sections allocated to each component, thereby enhancing code clarity and organization.

/* Hero */ CSS /* Hero end here*/

The project is characterized by responsiveness across diverse screen sizes. This adaptability is achieved through the incorporation of CSS @media queries, ensuring optimal functionality on a spectrum of devices, including phones, tablets, and laptops.



```css
@media screen and (max-width: 780px) {
    .Hero-container {
        display: block;
        margin: 0 auto;
        padding-top: 100px;
    }
}
```

Figure 41. CSS media queries.

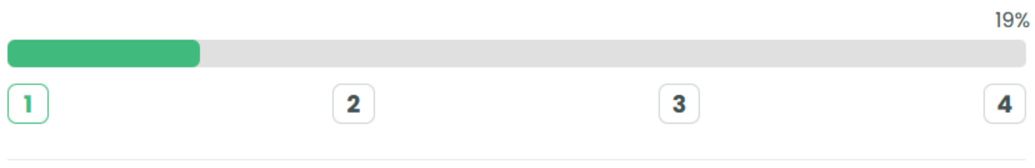
The thesis code could be fine in GitHub: github.com/samiwazni/MyPortfolio

6 Seven-Week Diary Entries.

6.1 Week One: September 18 – September 22

During this week, a new version of Financiar24.es's Step 2 Form was planned and implemented. The prior version, noted for its extensive length, offered an opportunity to enhance the user experience by facilitating a smoother progression. The solution involved a comprehensive restructuring of the form into three sections: Personal Data, Additional Data, and Financial Data, each dedicated to gathering information pertinent to its designated category.

To enhance the user experience, a progress bar was developed, providing users with a visual indication of their progress throughout the form. This progress bar gives users a clear sense of how far the process has advanced. Picture 9



Picture 9. Form progress bar.

For the development phase, a holistic combination of PHP, jQuery, and SCSS was employed. Within the PHP, three div elements were introduced, bearing the IDs of "stepone," "steptwo," and "stepthree," each associated with corresponding form sections. Within each section, an implemented validation mechanism was embedded to ensure the completion of all requisite inputs. If the validation process returns a positive result, users progress to the subsequent phase.

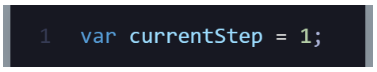
In Section One, a "Continue" button was provided. Section Two included both "Continue" and "Previous" buttons, enabling users to navigate forward and backwards as necessary. In Section Three, only a "Previous" button was made available. Finally, at the end of the form, a "Submit" button was included to transmit the collected information to the database. Whenever the buttons are pressed, a loading icon will briefly appear for one second. This redesign complete with the progress bar, aims to streamline the user experience and encourage more efficient form completion.

This comprehensive redesign, with the addition of a dynamic progress bar and the strategic use of jQuery, ensures a better user experience and makes form completion more efficient. jQuery was used to create features that dynamically track the length of form sections and guide users correctly when they click the 'Continue' or 'Previous' buttons. Additionally, another jQuery function was applied to validate user inputs, ensuring that all required fields are properly filled, simplifying the user's form filling.

6.1.1 Coding for Week One

Initialized variables for tracking the current form step, selecting form sections, calculating the total steps, setting an initial progress percentage, and selecting a loading icon. Set up a click event handler for elements with the class 'continue-button'. When clicked, it:

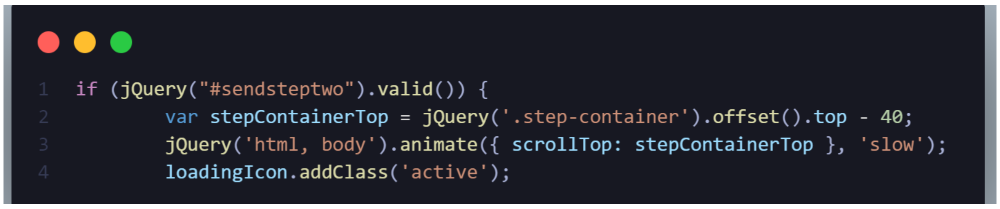
- Prevents the default form submission behavior.
- Validate the form with the ID 'sendsteptwo'.
- Scrolls to the top of an element with the class 'step-container' with an offset.
- Always start with the first step (currentStep === 1).
- After a 1-second delay:
    - Removes the loading icon's 'active' class.
    - Hides the current form section.
    - Shows the next form section.
    - Shows the 'previous-button'.
    - Adjust the progress bar's width and text(%).
    - Adds a 'completed' class to completed steps. var currentStep = 1;



```
var currentStep = 1;
```

Figure 42. Defined variable with a value.

When the form inputs are validated and the form advances to section two, a brief loading icon appears in the window. This gives the user the sense that an action is occurring within the form. After a short moment, the loading icon vanishes, and the following section becomes visible.

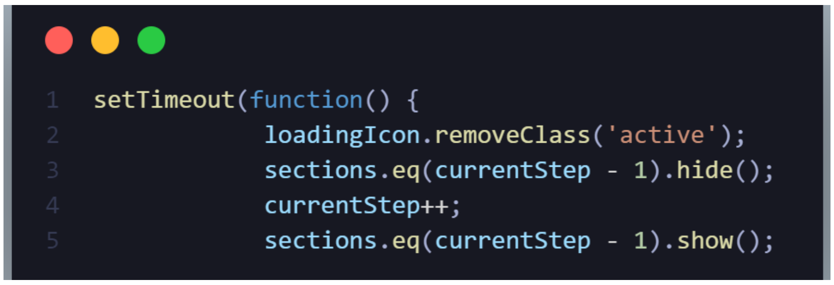


```
if (jQuery("#sendsteptwo").valid()) {
        var stepContainerTop = jQuery('.step-container').offset().top - 40;
        jQuery('html, body').animate({ scrollTop: stepContainerTop }, 'slow');
        loadingIcon.addClass('active');
```

Figure 43. jQuery to check the input validation and create a loading icon.

After that, the code, wrapped in a setTimeout function, does the following:

- Removes the 'active' class from a loading icon, likely hiding it.
- Hides the current section.
- Increments the current step.
- Shows the next section, making the form progress to the next step.



```
setTimeout(function() {
            loadingIcon.removeClass('active');
            sections.eq(currentStep - 1).hide();
            currentStep++;
            sections.eq(currentStep - 1).show();
```

Figure 44. jQuery code to hide the current section and show the next section.

After the form was deployed by Draivi's CTO, an A/B test was initiated to ensure the smooth operation of all code and identify potential issues that clients might encounter when using the form. The Microsoft Clarity tool was employed to analyze the form's behavior and gather essential analytics.

6.2 Week Two: September 25 – September 29

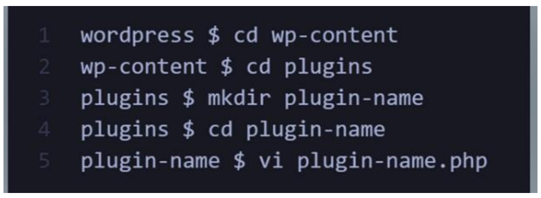
During this week, the focus was on planning and implementing a new theme for Draivi's Top5Credits.com sites, with particular attention to the components that interact with the site's Forms, Nav, Footer, Fonts, Cards, and related elements. The existing theme presented several issues, and the initiative began with restyling the SCSS for modern and user-friendly forms. A new site's font was added, the previous logo was replaced, and the new logo was aligned in both the navigation and footer sections. These efforts form an integral part of ongoing work aimed at enhancing the site's overall appearance and functionality.

During the same week, an initiation of a new project was undertaken. The objective of this project was to develop cookies for Draivi's websites, all of which are based on the WordPress platform. To accomplish this task, a specialized Plugin was designed, providing a convenient means of activating the requisite code and functionalities across multiple sites. This Plugin was crafted using a combination of PHP, jQuery, HTML and SCSS.

Within the context of WordPress directory structures, there exists a designated plugin folder, as illustrated in Picture 10. Within this pre-established structure, a file named "draivi-cookies.php" was created. In the file, all the plugin code lines were added and instructed to run the Plugin.

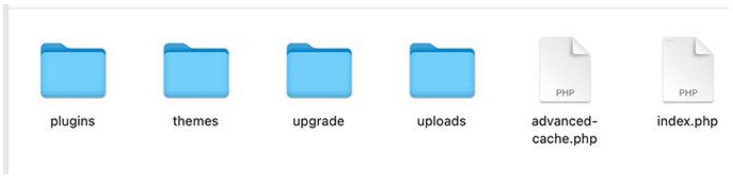
Roadmap on how to create the Plugin:

- Navigate to the WordPress installation’s wp-content directory.
- Open the plugins directory.
- Create a new directory and name it after the plugin (e.g. plugin-name).
- Create a new PHP file (e.g. plugin-name.php).



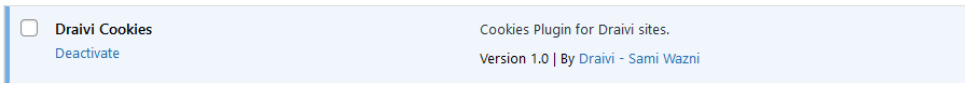
```
wordpress $ cd wp-content
wp-content $ cd plugins
plugins $ mkdir plugin-name
plugins $ cd plugin-name
plugin-name $ vi plugin-name.php
```

Picture 10. Create WordPress Plugin’s files.



Picture 11. WordPress file structures.

Upon the completion of the setup process, the Plugin will manifest itself within the Plugins section of the WordPress dashboard. To initiate the activation of the Plugin, it is imperative to select the "Active" option in Picture 12. This action grants permission to access and observe the ongoing development seamlessly.


Draivi Cookies
Deactivate

Cookies Plugin for Draivi sites.
Version 1.0 | By Draivi - Sami Wazni

Picture 12. Draivi's Cookies Plugin.

6.2.1 Coding for Week Two

With a concise set of code lines in 'draivi-cookies.php,' the Plugin becomes visible in the Plugins section of the WordPress dashboard. The integration of jQuery and CSS files ensures the utilization of JavaScript functionality (through jQuery) and styling capabilities (through SCSS) within the Plugin.

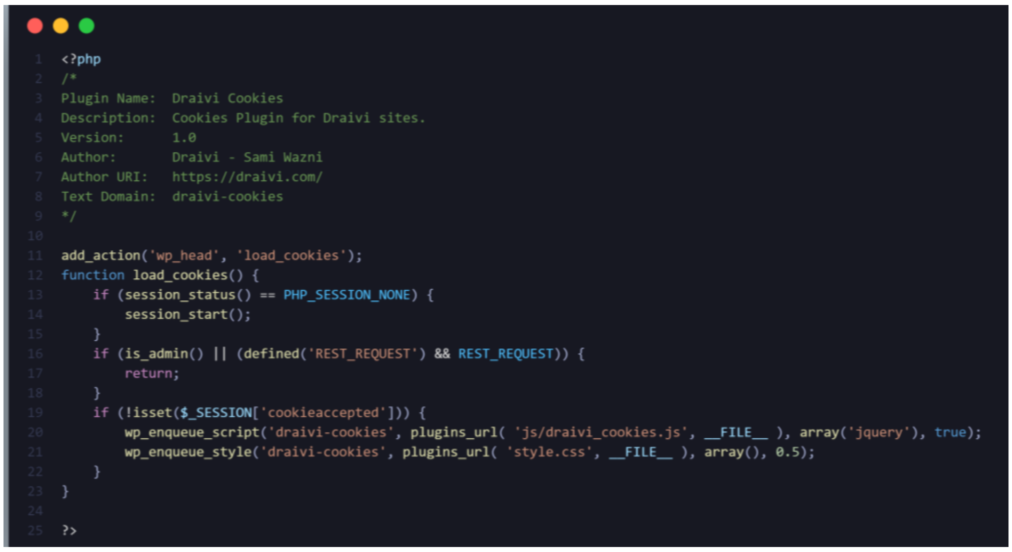

```php
<?php
/*
Plugin Name:  Draivi Cookies
Description:  Cookies Plugin for Draivi sites.
Version:      1.0
Author:       Draivi - Sami Wazni
Author URI:   https://draivi.com/
Text Domain:  draivi-cookies
*/

add_action('wp_head', 'load_cookies');
function load_cookies() {
    if (session_status() == PHP_SESSION_NONE) {
        session_start();
    }
    if (is_admin() || (defined('REST_REQUEST') && REST_REQUEST)) {
        return;
    }
    if (!isset($_SESSION['cookieaccepted'])) {
        wp_enqueue_script('draivi-cookies', plugins_url( 'js/draivi_cookies.js', __FILE__ ), array('jquery'), true);
        wp_enqueue_style('draivi-cookies', plugins_url( 'style.css', __FILE__ ), array(), 0.5);
    }
}

?>
```

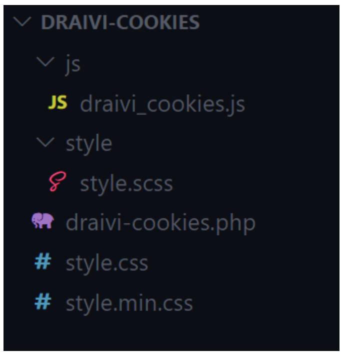
Figure 45. PHP code to create a Plugin.

6.3 Week Three: October 2 – October 6

During this week, the focus remained on advancing the Cookies Plugin. The structuring of folders and files, outlined in Picture 13 was aligned with user requirements. A pivotal step involved the creation of the main PHP file, adhering to the directive introduced in week 29, stipulating that the file should bear the Plugin's name. Consequently, the PHP file was designated as draivi-cookies.php, encompassing the entire backend code for the Plugin, along with an echo of the contents slated for display on the front-end.

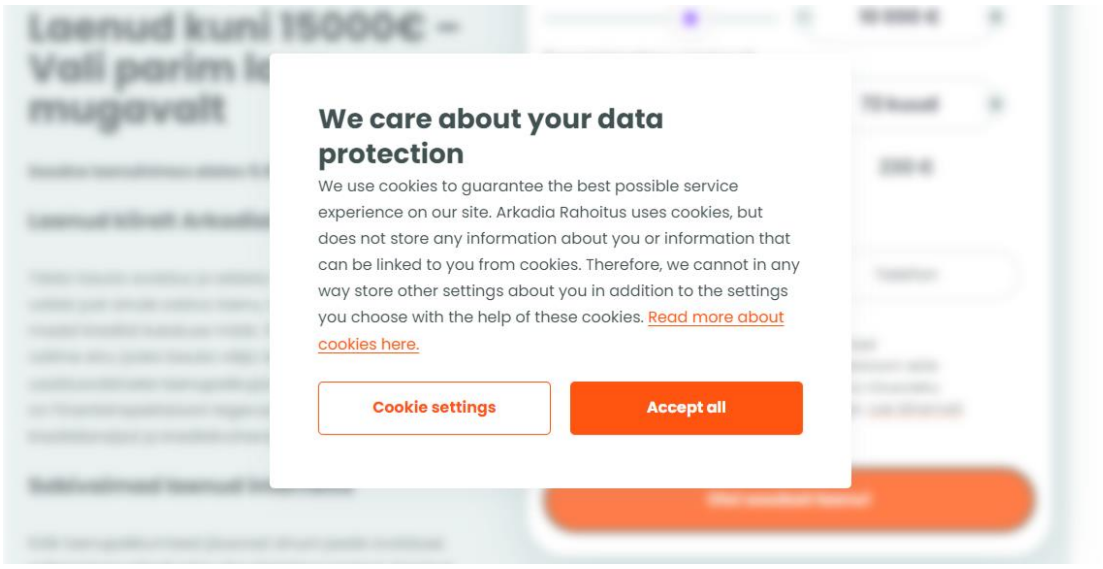
The front-end manifestation, denoted as Picture 14, featured two buttons—Accept All and Cookies Settings accompanied by explanatory text elucidating Draivi's cookies. A second window, depicted in Picture 15 dynamically surfaced upon the user's interaction with the Cookies Settings button. This window elucidated each cookie group and provided checkboxes, enabling users to select their preferred cookies for browser integration.

The functionality of the Plugin encapsulated in draivi-cookies.js, orchestrated various actions. Notably, the display of cookies commenced three seconds after the user-initiated page loading. Access to sites was contingent upon the user's acceptance of necessary cookies. Activating the Cookies Settings button revealed a settings window, where essential cookies were pre-checked. The primary functionality involved obstructing cookies and configuring them based on user preferences. Additionally, the styling nuances of the Cookies were curated within the style.scss file.



```
DRAIVI-COOKIES
  js
    draivi_cookies.js
  style
    style.scss
  draivi-cookies.php
  style.css
  style.min.css
```

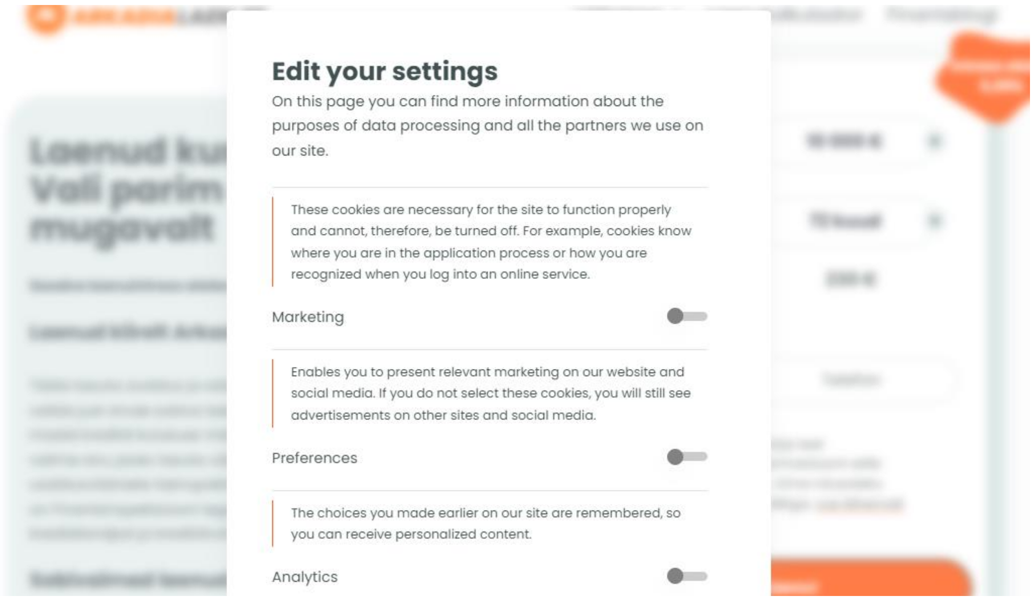
Picture 13. The Cookies Plugin structure files.



We care about your data protection

We use cookies to guarantee the best possible service experience on our site. Arkadia Rahoitus uses cookies, but does not store any information about you or information that can be linked to you from cookies. Therefore, we cannot in any way store other settings about you in addition to the settings you choose with the help of these cookies. Read more about cookies here.

Picture 14. The Cookies Plugin front window



Edit your settings

On this page you can find more information about the purposes of data processing and all the partners we use on our site.

These cookies are necessary for the site to function properly and cannot, therefore, be turned off. For example, cookies know where you are in the application process or how you are recognized when you log into an online service.

Marketing

Enables you to present relevant marketing on our website and social media. If you do not select these cookies, you will still see advertisements on other sites and social media.

Preferences

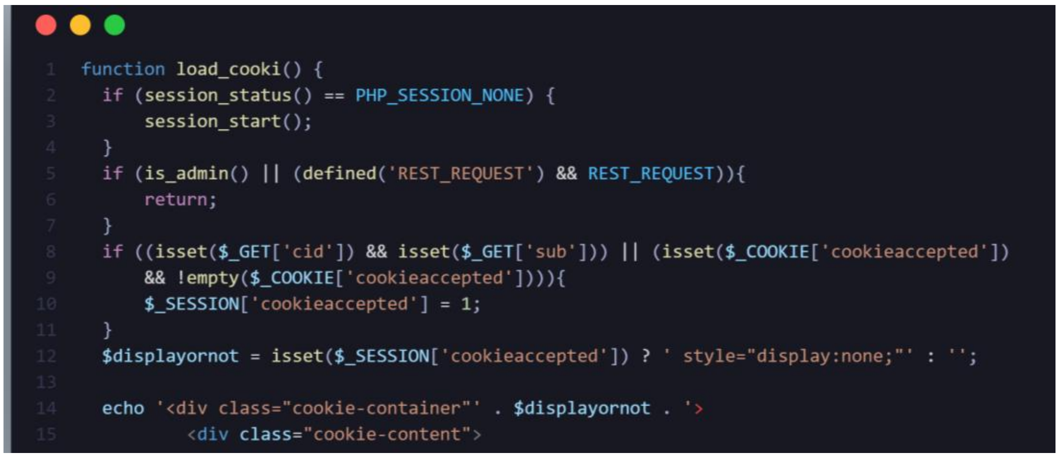
The choices you made earlier on our site are remembered, so you can receive personalized content.

Analytics

Picture 15. The Cookies settings

6.3.1 Coding for Week Three

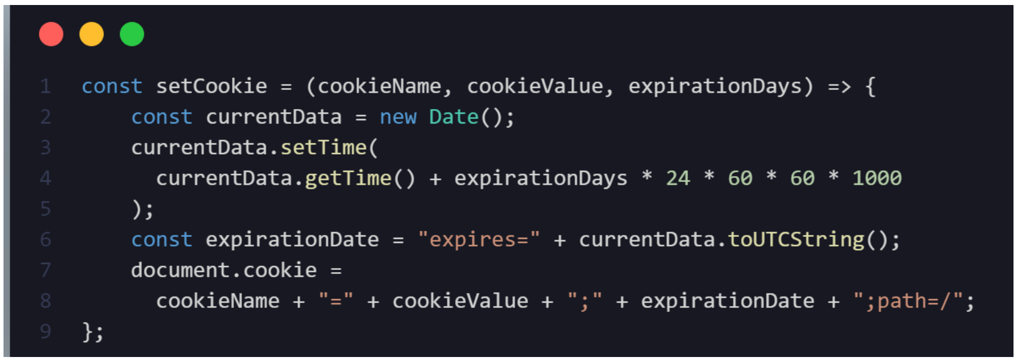
This code was for handling cookie acceptance on a website, controlling the display of a cookie container based on user actions and session status. Figure 46.



```
function load_cooki() {
  if (session_status() == PHP_SESSION_NONE) {
      session_start();
  }
  if (is_admin() || (defined('REST_REQUEST') && REST_REQUEST)){
      return;
  }
  if ((isset($_GET['cid']) && isset($_GET['sub'])) || (isset($_COOKIE['cookieaccepted'])
      && !empty($_COOKIE['cookieaccepted']))){
      $_SESSION['cookieaccepted'] = 1;
  }
  $displayornot = isset($_SESSION['cookieaccepted']) ? ' style="display:none;"' : '';

  echo '<div class="cookie-container"' . $displayornot . '>
          <div class="cookie-content">
```

Figure 46. jQuery code for handling cookie acceptance on a website.

This jQuery function is built for the Cookies Plugin which simplifies the process of setting a cookie by taking the cookie's name, value, and expiration days as parameters and configuring the necessary attributes. Figure 47.



```
const setCookie = (cookieName, cookieValue, expirationDays) => {
    const currentData = new Date();
    currentData.setTime(
      currentData.getTime() + expirationDays * 24 * 60 * 60 * 1000
    );
    const expirationDate = "expires=" + currentData.toUTCString();
    document.cookie =
      cookieName + "=" + cookieValue + ";" + expirationDate + ";path=/";
};
```

Figure 47. jQuery code for the process of setting a cookie.

6.4 Week Four: October 9 – October 13

During this week, the new theme was published/activated on Draivi’s Top5Credits sites, spanning six languages. After several weeks of development, the theme underwent testing on various devices, including IOS, Android, Windows, and Mac. Following activation, the logos in the headers and footers were updated, and all associated files, including email templates, were modified to reflect the new logo.

Simultaneously, additional functionality was integrated into Draivi’s Cookies Plugin. This feature conducts a user agent check to distinguish between human users and web crawlers (bots), functioning as follows:

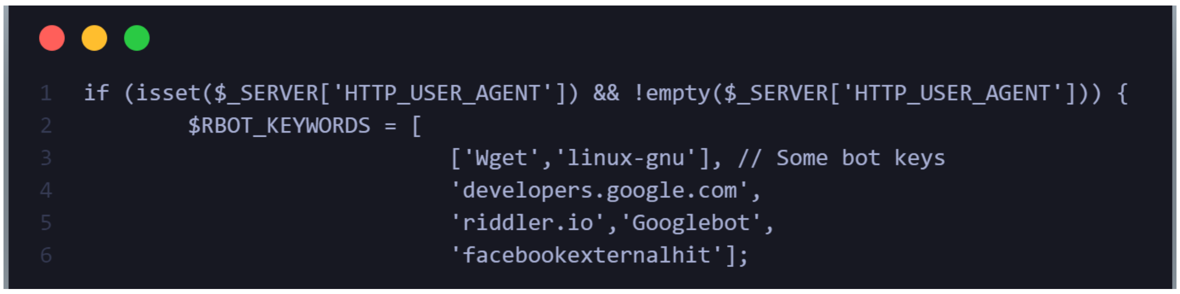
User Agent Check: The "User-Agent" header in HTTP requests contains information about the client, such as browser type and version. Bots and web crawlers often have distinct user agent strings.

Bot Identification: An array, $RBOT_KEYWORDS Figure 48 was implemented, containing keywords or patterns commonly found in bot user agent strings. This serves as a list of known bot identifiers.

Bot Detection: The code examines the "User-Agent" header of incoming requests for matches with the specified bot-related keywords or patterns. A match indicates that the request is likely from a bot or web crawler.

Early Return for Bots: If the code identifies a request with a user agent matching any specified bot keywords, it exits the function early (using return). This avoids executing the cookie management logic for that particular request, optimizing efficiency.

The purpose of this code is to enhance the plugin's efficiency by avoiding unnecessary cookie processing for bot requests. Bots typically do not require the same cookie handling as human users. By bypassing cookie management for bots, server resources are conserved, reducing unnecessary workload.



```
if (isset($_SERVER['HTTP_USER_AGENT']) && !empty($_SERVER['HTTP_USER_AGENT'])) {
        $RBOT_KEYWORDS = [
                                ['Wget','linux-gnu'], // Some bot keys
                                'developers.google.com',
                                'riddler.io','Googlebot',
                                'facebookexternalhit'];
```

Figure 48. PHP code shortcode for bots.

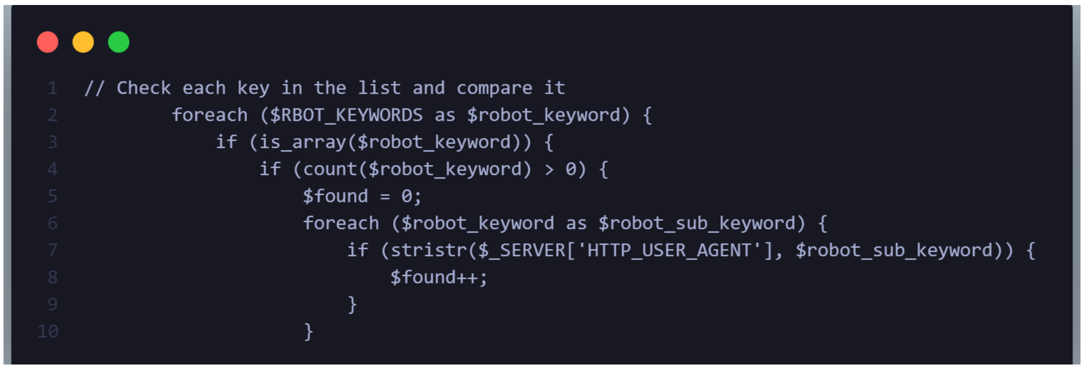

```
// Check each key in the list and compare it
        foreach ($RBOT_KEYWORDS as $robot_keyword) {
            if (is_array($robot_keyword)) {
                if (count($robot_keyword) > 0) {
                    $found = 0;
                    foreach ($robot_keyword as $robot_sub_keyword) {
                        if (stristr($_SERVER['HTTP_USER_AGENT'], $robot_sub_keyword)) {
                            $found++;
                        }
                    }
```

Figure 49. PHP code to manage displaying the cookies for humans and bots.

Another function prevents the Plugin from displaying on the terms site, allowing users to read the Cookie terms and site information. An information page was enhanced with a button enabling users to reopen the Cookies settings and adjust their preferences. Debugging was initiated for the Cookies' new functionality related to the User Agent. Various testing methods were employed, such as the Google Extension User Agent Switcher, allowing the user agent of the browser to be changed. Alternatively, the DevTool of Google Chrome was used, selecting "Network condition" from the hamburger menu under site inspect. Setting the User Agent as a bot prevents the Plugin from executing, and reverting to the default user agent allows the Plugin to execute and load all functionalities.

6.5 Week Five: October 16 – October 20

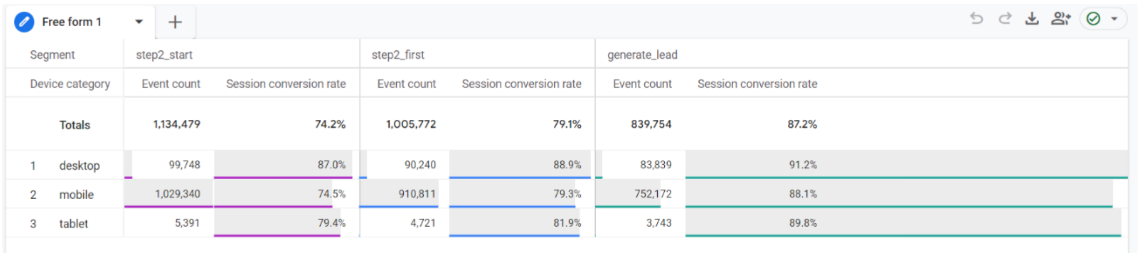
During this week, a new feature for the Komparate.com site was planned. Komparate is an electricity price comparison platform that calculates live electricity prices. The planned feature involves the calculation and implementation of monthly hourly averages for electricity prices. The process begins by obtaining the sum of monthly prices, followed by the sum of hourly prices, resulting in the calculation of the hourly average per month. Subsequently, a data table is generated, spanning 12 months, and detailing hourly intervals from 0-1, 2-3, and so forth, serving as a statistical table. This table resembles the one below but includes additional content.

Table 1. Electricity average price.

Month	0-1	2-3	etc.
Jan	8.23	7.45	0.00
Feb	5.48	6.78	0.00

During the same week, the focus was on utilizing Google Analytics 4 to analyze user data on the Financiar24 site. Initially, there was an older version of the form, which was subsequently replaced with a new version featuring enhanced features to improve the overall user experience. To assess the effectiveness of the new form and determine its impact on user experience, an A/B testing approach was implemented.

To gather relevant data for the A/B testing and evaluate the progress made, Google Analytics 4 was employed. The data obtained was organized into a table for both versions of the form, and graphical representations were generated for visual analysis of the results. A thorough comparison was then conducted between the two sets of data. The findings indicated that the new form version yielded superior results, leading to the decision to conclude the testing phase and officially implement the new and improved form version. Picture 16

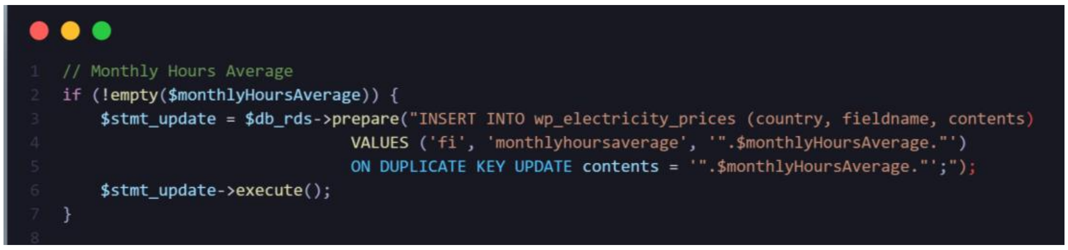


Segment	step2_start		step2_first		generate_lead	
Device category	Event count	Session conversion rate	Event count	Session conversion rate	Event count	Session conversion rate
Totals	1,134,479	74.2%	1,005,772	79.1%	839,754	87.2%
1 desktop	99,748	87.0%	90,240	88.9%	83,839	91.2%
2 mobile	1,029,340	74.5%	910,811	79.3%	752,172	88.1%
3 tablet	5,391	79.4%	4,721	81.9%	3,743	89.8%

Picture 16. Google Analytics 4 statistic table to read the forms data.

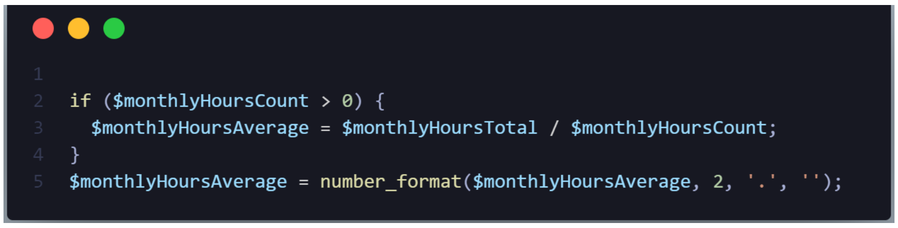
6.5.1 Coding for Week Five

This code creates a table to collect data and inserts it into the database for calculating the monthly average hours. Picture 17.



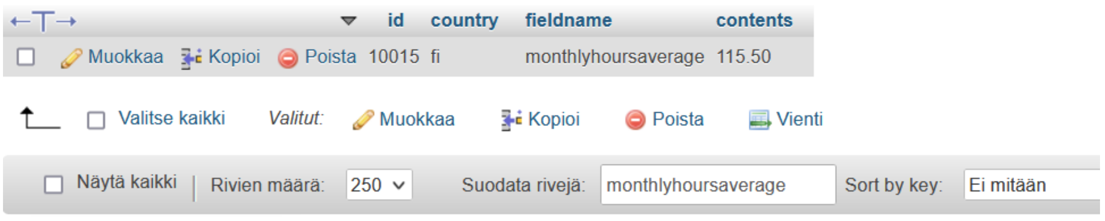
```
// Monthly Hours Average
if (!empty($monthlyHoursAverage)) {
    $stmt_update = $db_rds->prepare("INSERT INTO wp_electricity_prices (country, fieldname, contents)
                                    VALUES ('fi', 'monthlyhoursaverage', '".$monthlyHoursAverage."')
                                    ON DUPLICATE KEY UPDATE contents = '".$monthlyHoursAverage."';");
    $stmt_update->execute();
}
```

Figure 50. PHP code to create a table in the database.



```

if ($monthlyHoursCount > 0) {
  $monthlyHoursAverage = $monthlyHoursTotal / $monthlyHoursCount;
}
$monthlyHoursAverage = number_format($monthlyHoursAverage, 2, '.', '');
```

Figure 51. PHP code to calculate the average.



	id	country	fieldname	contents
☐ Muokkaa Kopioi Poista	10015	fi	monthlyhoursaverage	115.50

Picture 17. Database table for calculating the electricity.

6.6 Week Six: October 23 – October 27

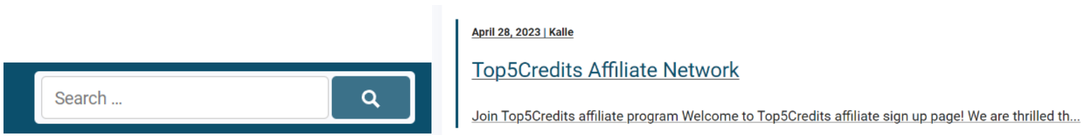
During this week, the focus was on planning and implementing a new feature for Draiv's Top5credits site. This feature involves the introduction of a search bar and a corresponding function for processing search queries. The initial step included the creation of the UI design, as illustrated in Picture 18.

The search functionality comes with specific requirements. Notably, the search button remains inactive if the input field is empty. Additionally, the search results exclude noindex pages, such as the Thank You and Forms pages. A key aspect of this feature is the implementation of a clean URL structure, ensuring clarity when users search for a word (e.g. "url/search/category/searched-word"). The placement of the search bar should be in the site's footer.

To address these requirements, a dedicated file, search.php, was created to encapsulate the necessary functionality. The functions within this file accomplish the following tasks:

- Validate the input value and conduct the search accordingly.
- Redirect the results to the designated search result page.
- Display key information on the result page, including the header name, date, and a concise paragraph.
- Implement a character limit function for the short paragraph, displaying only the first 100 characters followed by an ellipsis (...).
- Enable clickable results, allowing users to navigate to the result page by selecting a result.

This strategic implementation aims to enhance the user experience by providing a robust and intuitive search feature on Draiv's Top5credits site.



April 28, 2023 | Kalle

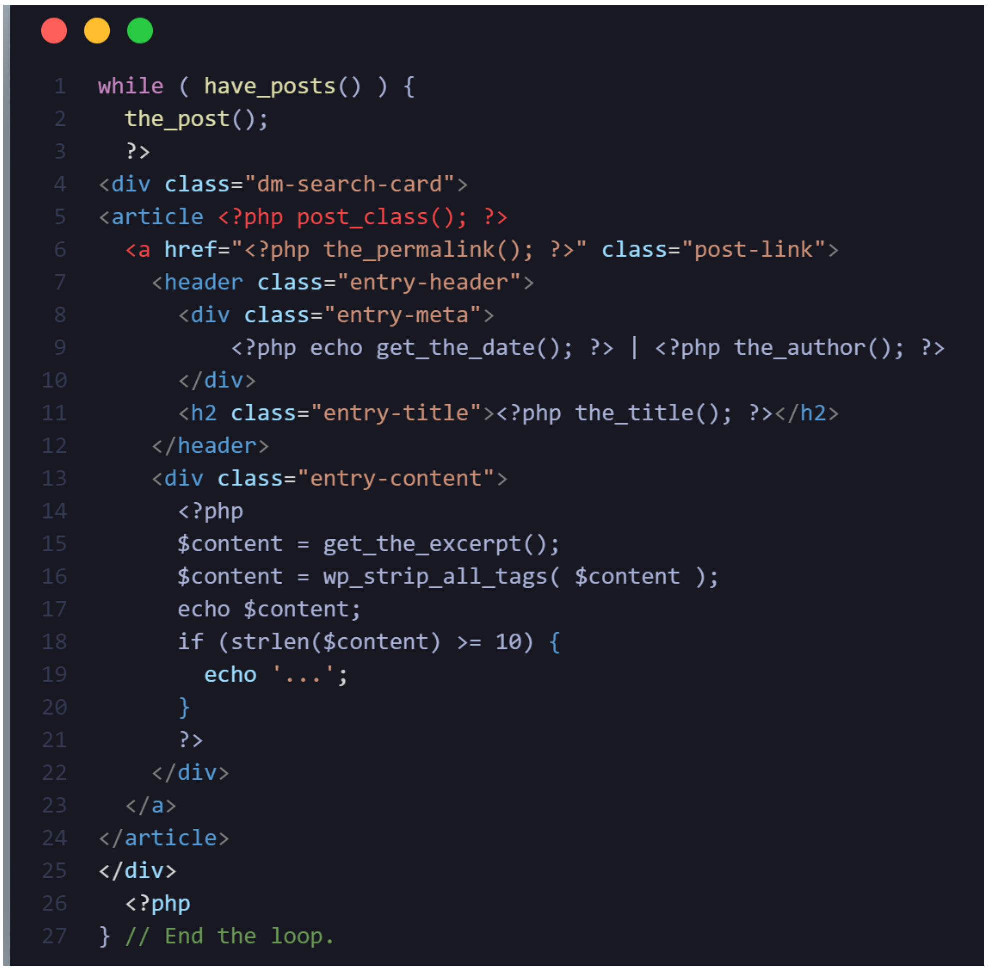
Top5Credits Affiliate Network

Join Top5Credits affiliate program Welcome to Top5Credits affiliate sign up page! We are thrilled th...

Picture 18. Search bar and search result.

In Figure 52 below, the code is within a loop, likely iterating through a list of the posts using the `have_posts()` and `the_post()` functions. For each post, it generates a card-like structure for a search result, enclosed in a `div` element with the class "dm-search-card.

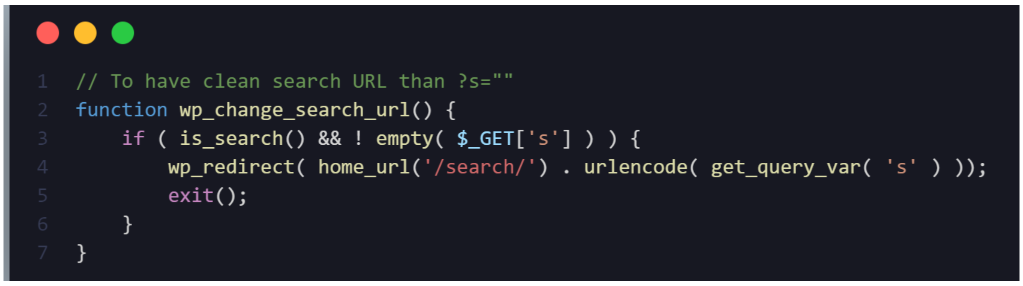
6.6.1 Coding for Week Six



```php
while ( have_posts() ) {
  the_post();
  ?>
<div class="dm-search-card">
<article <?php post_class(); ?>
  <a href="<?php the_permalink(); ?>" class="post-link">
    <header class="entry-header">
      <div class="entry-meta">
          <?php echo get_the_date(); ?> | <?php the_author(); ?>
      </div>
      <h2 class="entry-title"><?php the_title(); ?></h2>
    </header>
    <div class="entry-content">
      <?php
      $content = get_the_excerpt();
      $content = wp_strip_all_tags( $content );
      echo $content;
      if (strlen($content) >= 10) {
        echo '...';
      }
      ?>
    </div>
  </a>
</article>
</div>
  <?php
} // End the loop.
```

Figure 52. Loop for the search card

Figure 53 below defines a function named `wp_change_search_url`. This function is likely intended to modify the default search URL behavior on a WordPress website. Here's a breakdown of what the code does:



```php
// To have clean search URL than ?s=""
function wp_change_search_url() {
    if ( is_search() && ! empty( $_GET['s'] ) ) {
        wp_redirect( home_url('/search/') . urlencode( get_query_var( 's' ) ));
        exit();
    }
}
```

Figure 53. PHP code for modifying the default search URL.

This function is designed to alter the search URL structure on a WordPress site. When a search is performed, it redirects the user to a URL where the search query is included in the path after "/search/", ensuring a modified search URL format.

6.7 Week Seven: October 30 – November 3

During this week, various tasks were undertaken. One of the primary objectives was to update all of Draivi's websites with the new brand logo from Top5Credits sites. This logo was integrated not only into the email templates but also into landing pages and loan offer cards. It was imperative to ensure a consistent implementation of the new logo version across all these elements.

Simultaneously, the Nätfinans.se site underwent several updates. This involved the modification of the single-page hero post, adjusting text colors, aligning images, ensuring consistent site width, and overall enhancing visual coherence.

In addition to these tasks, the Draivi development team activated a review plugin. Upon activation, a schema setup was configured to enhance Google search visibility, aiming for optimal results. To achieve this, an investigation into the entire system was conducted, including understanding the relationships between the comparison tool and its database table, the review Plugin and its database table, and the Tables.php in the main site’s file.

Currently, the review plugin loads with the tag-name of the correct provider, and the name is retrieved from a tool responsible for collecting and managing all reviews. However, this structure poses a risk and needs to be improved. If the out-name is changed for any reason, the entry no longer aligns with the correct review, resulting in reviews disappearing.

To mitigate this risk, an alternative approach was considered. Instead of using the out-name with the shortcode, the "id" of the entry in the shortcode (e.g. "wp_comparison_fin") would be defined. This "id" information would be stored in the reviews database (e.g. "wp_reviews_fin"), along with the page path.

The steps taken to address this issue, initially at the local level, included:

- Adding missing fields in the review table (comparison_id and page_path).
- Incorporating in the Comparison tool the display of the entry's ID, currently not visible.
- Potentially providing a shortcode example for the review directly in the Comparison tool.
- Implementing functionality to store the page_path and comparison_id when submitting a review.
- Populating the existing entries in the review tables with correct paths and IDs.
- Transitioning to use the paths for fetching existing reviews instead of the tag-name.

7 Conclusion

In conclusion, the significance of front-end development in web development cannot be overstated, as it plays a pivotal role in shaping the functionality and visual appeal of websites and applications. The front-end, or client side, bears the responsibility of crafting user interfaces, optimizing for performance and accessibility, ensuring a seamless user experience, and collaborating seamlessly with the back end or server side. This collaboration is essential for maintaining alignment between the front-end presentation and design, as well as synchronizing with back-end functionality.

Throughout this thesis, a comprehensive exploration of front-end fundamentals has been presented. This exploration equips readers with a clear understanding of how various front-end tools synergize to produce the end result of a website or web application tailored to diverse needs. The culmination of this work manifests in a fully functional React-based portfolio application. This application, available freely on GitHub, comprises six components: Home, Nav, Skills, Contact, About, and Projects.

Noteworthy features of the application include the capability to send emails seamlessly through EmailJS. Leveraging React packages, the application streamlines this process, incorporating validation measures to ensure essential fields are consistently filled. By providing this React-based portfolio as an open and modifiable resource, this thesis seeks to contribute to the collective knowledge and ongoing innovation within the realm of front-end development.

References

Andrade, C.D.B., 2020. Web application for the Mentoring Academy program (Doctoral dissertation). https://bibliotecadigital.ipb.pt/handle/10198/22800

Dziallas, M., 2020. How to evaluate innovative ideas and concepts at the front-end?: A front-end perspective of the automotive innovation process. Journal of Business Research, 110, pp.502-518. https://www.sciencedirect.com/science/article/abs/pii/S0148296318302339

Hwangbo, H. and Lee, H., 2008, October. Reusing of information constructed in HTML documents: A conversion of HTML into OWL. In 2008 International Conference on Control, Automation and Systems (pp. 871-875). IEEE. https://ieeexplore-ieee-org.ezproxy.turkuamk.fi/document/4694654

Jazayeri, M., 2007, May. Some trends in web application development. In Future of Software Engineering (FOSE'07) (pp. 199-213). IEEE. https://ieeexplore.ieee.org/abstract/document/4221621

Pan, L., & Ma, J. S. (2022). HTML+CSS Implementation based on Image Intelligent Scene Recognition Algorithm. Proceedings - International Conference on Augmented Intelligence and Sustainable Systems, ICAISS 2022, 532–536. https://doi.org/10.1109/ICAISS55157.2022.10011034

Roldán, C.S., 2023. React 18 Design Patterns and Best Practices: Design, build, and deploy production-ready web applications with React by leveraging industry-best practices. https://www.scholarvox.com/catalog/book/88946133

Sengupta, D., Singhal, M. and Corvalan, D., 2016. Getting Started with React. Packt Publishing Ltd. https://books.google.fi/books?hl=en&lr=&id=-dzJDAAAQBAJ&oi=fnd&pg=PP1&dq=React&ots=kwxF3eVdsN&sig=Ru1IkL_Ha9ICcoRYIafjDX3hMIE&redir_esc=y#v=onepage&q=React&f=false

Xiao, Y., 2016. Web front-end architecture with Node js. platform. https://www.theseus.fi/handle/10024/114085