FRONT-END WEB DEVELOPMENT

Practical Training Report

Submitted in partial fulfillment for the award of degree of

Bachelor of Technology

in

Department of Computer Science & Engineering

Submitted to:
Mr. Pramod Nagar
Assistant Professor
Department of Computer Science

Submitted By:
Manvendra Singh
Roll No. 17/468
Univ. Roll No.: 17EUCCS037
Semester/Year: V Sem/III Year

Department of Computer Science & Engineering

University Department,

Rajasthan Technical University, Kota

January-2023

CERTIFICATE

ACKNOWLEDGEMENT

I take this opportunity to express a deep sense of gratitude towards Mr. Ishwar Singh Solanki, for providing excellent guidance, encouragement and inspiration through-out the practical training. Without his invaluable guidance, this practical training report would never have been a successful one.

I also want to give thanks to all the faculty members of department of Computer Science Engineering and my family for their love, support and encouragement during this time. I want to thank my dear friends, the ones that remain here and the ones that have already left, for making this period such an extraordinary experience in my life.

Last but not least, my heartiest thanks to all the persons, who have been a pillar of support during the arduous time of training.

Manvendra Singh

PREFACE

The present report is the outcome of the offline practical training on "**Front End Web-Development**" provided by "**Bit to Byte Robotics**". The objective of this training is to learn about Front End Web Development with practical applications and to get a masterful grip on it. This course provided me with hands-on experience and exposure to developing Front-End applications for browsers. This course also helped in building a strong foundation on Front-End which provided me with the tools to build a responsive web application.

During the tenure of 45 days, I learned about and worked on different technologies like HTML, CSS, JavaScript and GitHub. I have also built a Front-End project using these technologies.

This report explains the concepts learnt during the internship along with the description of how these technologies are used for the creation of the project.

CONTENTS

CHAPTER 1: INTRODUCTION ABOUT THE COURSE **1-2**
1.1 Objectives 1
1.2 Motivation 1
1.3 About the company 2
1.4 Layout of the Report 2
CHAPTER 2: BASICS OF FRONT-END WEB DESIGN **3**
CHAPTER 3: HTML **4-8**
3.1 Identifying the parts that make up an HTML tag 4
3.2 Determining when to use specific HTML tags 5
3.3 Correctly structuring nested HTML content 8
CHAPTER 4: CSS **9-12**
4.1 Introduction about CSS 9
4.2 Identifying the benefit of separating style from content 9
4.3 Using CSS to style a website 9
4.4 Structure of CSS 10
4.5 Targeting things in CSS 10
4.6 Some basic properties of CSS 11
CHAPTER 5: JAVASCRIPT **13-18**
5.1 Introduction to JavaScript 13
5.2 JavaScript Language Basics 13
5.3 JavaScript Loops, Condition 16
5.4 JavaScript Objects 16
5.5 Basics of ES6 17
5.6 JavaScript DOM 18
CHAPTER 6: PROJECT **19**
CONCLUSION **20**
REFERENCES **21**

List of Figures

Figure No.	Name of Figure	Page No.
3.1	Structure of an HTML webpage	5
4.1	Linking HTML and CSS file using link tag	9
4.2	Structure of CSS	10
6.1	Wordle Project made using HTML, CSS and JavaScript	19

List of Tables

Table No.	Name of Table	Page No.
3.1	Basic HTML tags	6
3.2	Formatting tags	6-7
3.3	Forms and Input tags	7
3.4	Images tags	7
3.5	Links tags	8
3.6	Lists tags	8
3.7	Tables tags	8
5.1	Data Types in JavaScript	14
5.2	Object Properties in JavaScript Example	16
6.2	Object Methods in JavaScript Example	17

CHAPTER 1: INTRODUCTION ABOUT THE COURSE

1.1 Objectives

Practical training is an integral part of skill development during the course of a technical degree. In practical training, the student works in the actual industry while gaining real life experiences and learning about the practical applications of the concepts learnt during the course of his/her technical degree. Practical training offers the students a platform for recognizing their interests and skills and refining them to absolute perfection. Practical training provides an opportunity to gain first-hand experience of the industry while working with other experiences engineering professionals and learning from them. Hence, practical training is of paramount importance for increasing the overall employability of student by increasing and refining his/her competence in the relevant skills.

For the particular course I pursued in the field of Front-End Web Development, the objectives were as follows:

- To learn about the basics of HTML, CSS, JavaScript, GitHub and how to use them in combination to design a powerful web user interface.
- To learn about the various job opportunities in the field.
- To learn about the use of git which is the industry standard for branch control.
- To learn about the methods to build static and dynamic web pages using HTML, CSS, and JS and optimizing them for web.

1.2 Motivation

The motivation of pursuing the field of Front-End Web Development for practical training was due to my actual interest in the field. Moreover, front-end requires the use of creativity and design talents which is in-line with my actual skillset. I wanted to learn more about the process of designing a user-interface. Before joining the course, I was already familiar with the basics of web design but the training course helped me in refining my skills to the level required by the industry. Due to this practical training, I was able to design a project using HTML, CSS and JS.

1.3 About the company

BBR is an emerging start-up Company providing innovative products to increase the productivity, safety and quality of life based on latest technologies like- Embedded Robotics, AI and IoT. The products are mainly developed for farmers to solve lethal and unsolved problems due to traditional equipment and technologies but we have also a range of hardware and software both type of products for industries and home appliance to replace traditional inefficient and risky system to increase the productivity and safety.

1.4 Layout of the Report

This layout contains the key points of this report.

- Chapter 2 describes the basics of front-end web design.
- Chapter 3 describes the various concepts and application of HTML in designing the web page.
- Chapter 4 describes how CSS is used to provide design and describing presentation of a webpage made in HTML
- Chapter 5 describes the use of JavaScript in making a webpage dynamic. It is the programming language of the web
- Chapter 6 describes the project created during the internship by combing the technologies learned during the span of 45 days of the practical training.

CHAPTER 2: BASICS OF FRONT-END WEB DESIGN

Front-End is the client-side graphical user interface of the website that provides a way for the user to interact with the website. Front-End is what the users usually see when they open a website.

Front-End Development is the development of the graphical user interface of a website, through the use of HTML, CSS and JavaScript, so that users can view and interact with that website.

Front-End Developer is responsible for creating the user interface of a website. A front-end developer defines how a web page will look to the end user.

Tools used in Front-End Development

- **HTML:** HyperText Markup Language (HTML) is the foundation of any website development process, without which a web page would not exist. Hypertext refers to text that contains links, also known as hyperlinks. A user will be taken to another web page when they click on a word or phrase with a hyperlink. Using a markup language, text can be represented as graphics, tables, links and other things. The overall structure of the website is provided by the HTML code.
- **CSS:** Cascading Styles Sheets (CSS) manages the site's display, enables you to give your site a distinctly individual design. It accomplishes this by storing style sheets that are triggered by other inputs, such as the screen size and resolution of the device, and sit on top of existing style rules. CSS can be added externally, internally, or as an HTML tag embedding.
- **JavaScript:** JavaScript is an event-based imperative programming language (as opposed to HTML's declarative language model) which is used to convert a static HTML page into a dynamic interface. The Document Object Model (DOM), included in the HTML standard, can be used by JavaScript code to modify a webpage in response to events like user input. JavaScript code may actively retrieve content from the web using the AJAX technique (in addition to the traditional HTML page retrieval), and it can also respond to server-side events, giving web pages a truly dynamic feel.

CHAPTER 3: HTML

HyperText Markup Language (HTML) is the standard markup language for documents that are designed to be displayed in a web browser. Technologies like Cascading Style Sheets (CSS) and scripting languages like JavaScript can be integrated with HTML to design a powerful web interface.

HTML elements form the foundation of HTML pages. By indicating structural semantics for text elements like headings, paragraphs, lists, links, quotations and other objects, HTML offers a way to generate structured texts. Tags, which are written in angle brackets, are used to distinguish HTML elements.

3.1 Identifying the parts that make up an HTML tag

An HTML tag consists of three main parts:

- Opening tag
- Content
- Closing tag

Web browser typically read the HTML document from top to bottom and left to right. HTML tags are used for the initial creation of HTML documents and to assign their properties/attributes.

An HTML file must have some essential tags the HTML file can be distinguished from the simple text file by the web browser. There is no upper limit on the number of tags that can be used in an HTML file.

All HTML tags must be enclosed within < > these brackets.

With a few exceptions, if open tag <tag> is used, then its corresponding close tag must also be used </tag>.

Syntax of an HTML tag:

```
<tagname>Content</tagname>
```

Structure of an HTML webpage: The basic structure of a typical HTML page is shown below. It contains some essential tags that form the foundation of every single HTML page.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title></title>
</head>
<body>

</body>
</html>
```

Figure 3.1: Structure of an HTML webpage

<!DOCTYPE HTML>: Every HTML document should start with this line. It indicates to the browser that the document is a modern HTML webpage.

<html>: This tag indicates the start and end of the HTML document. Everything in the webpage goes inside this tag.

<head>: This tag is a container for the metadata and is placed between the <html> tag and the <body> tag. Metadata is data about the HTML document which is not displayed on the screen. It typically defines the document title, character set, styles, scripts and other meta information. It controls and defines the HTML document.

<meta>: This tag tells the browser about the kind of characters that are used in the document. utf-8 is most commonly used as it allows many languages in the world.

<title>: This tag defines the title of the webpage that is shown in the title bar of the web browser. This title is also shown in search results as the link.

<body>: This tag defines the body of the HTML document. This includes the content that is actually rendered on the screen in a website.

3.2 Determining when to use specific HTML tags

HTML tags should be used according to their specific functions when required as per the requirement of the webpage.

The following are the different categories of HTML tags:

- Basic HTML

Tag	Description
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	Contains metadata/information for the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a thematic change in the content
<!--…-->	Defines a comment

Table 3.1: Basic HTML tags [4]

- Formatting

Tag	Description
<abbr>	Defines an abbreviation or an acronym
<address>	Defines contact information for the author/owner of a document/article
<b>	Defines bold text
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<bdo>	Overrides the current text direction
<blockquote>	Defines a section that is quoted from another source
<cite>	Defines the title of a work
<code>	Defines a piece of computer code
<del>	Defines text that has been deleted from a document
<dfn>	Specifies a term that is going to be defined within the content
	Defines emphasized text
<i>	Defines a part of text in an alternate voice or mood
<ins>	Defines a text that has been inserted into a document
<kbd>	Defines keyboard input
<mark>	Defines marked/highlighted text
<meter>	Defines a scalar measurement within a known range (a gauge)
<pre>	Defines preformatted text
<progress>	Represents the progress of a task
<q>	Defines a short quotation
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<s>	Defines text that is no longer correct
<samp>	Defines sample output from a computer program
<small>	Defines smaller text
	Defines important text

<sub>	Defines subscripted text
<sup>	Defines superscripted text
<template>	Defines a container for content that should be hidden when the page loads
<time>	Defines a specific time (or datetime)
<u>	Defines some text that is unarticulated and styled differently from normal text
<var>	Defines a variable
<wbr>	Defines a possible line-break

Table 3.2: Formatting tags [4]

- Forms and Input

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<button>	Defines a clickable button
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<datalist>	Specifies a list of pre-defined options for input controls
<output>	Defines the result of a calculation

Table 3.3: Forms and Input tags [4]

- Images

Tag	Description
<img>	Defines an image
<map>	Defines a client-side image map
<area>	Defines an area inside an image map
<canvas>	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content
<picture>	Defines a container for multiple image resources
<svg>	Defines a container for SVG graphics
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<datalist>	Specifies a list of pre-defined options for input controls
<output>	Defines the result of a calculation

Table 3.4: Images tags [4]

- Links

Tag	Description
<a>	Defines a hyperlink
<link>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<nav>	Defines navigation links

Table 3.5: Links tags [4]

- Lists

Tag	Description
	Defines an unordered list
	Defines an ordered list
	Defines a list item
<dl>	Defines a description list
<dt>	Defines a term/name in a description list
<dd>	Defines a description of a term/name in a description list

Table 3.6: Lists tags [4]

- Tables

Tag	Description
<table>	Defines a table
<caption>	Defines a table caption
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table
<col>	Specifies column properties for each column within a <colgroup> element
<colgroup>	Specifies a group of one or more columns in a table for formatting

Table 3.7: Tables tags [4]

3.3 Correctly structuring nested HTML content

The main rule to follow when using nested HTML content is: The tag opened most recently is always the following tag to close. Improper HTML Tag nesting will cause a visual design page break due to the browser's incapability to render (read) tags that are out of place. Other errors may occur. Other than to avoid visual webpage errors, HTML nesting is used for CSS (Cascading Style Sheets). When tags are nested, CSS relies on the HTML attributes classes and id to identify where to apply the style, marked by the opening tag and by the closing tag.

CHAPTER 4: CSS

4.1 Introduction about CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. A style sheet language is a computer language that expresses the presentation of structured documents.

4.2 Identifying the benefit of separating style from content

The purpose of CSS is to make it possible to separate content from presentation, including layout, colours, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve page load speed between the pages that share the file and its formatting.

The ability to offer the same HTML page in several styles for various rendering techniques, including on-screen, in print, by voice (using a speech-based browser or screen reader), and on Braille-based tactile devices, is also made possible by the separation of formatting and content. If a user accesses the material on a mobile device, CSS additionally provides rules for different formatting.

4.3 Using CSS to style a website

To use CSS to style a website, we first need to create a new file and save it with the extension .css. Moreover, it's generally preferred to put the CSS file into a css folder for better organization.

After creating and saving the CSS file, we just need to link the CSS and HTML files together so that the browser can locate the CSS file.

To link the HTML file and the CSS file, add a link tag in the head of the HTML document:

```
<head>
    ⋮
    <link href="css/main.css" rel="stylesheet">
</head>
```

Figure 4.1: Linking HTML and CSS file using link tag

The link element is used to link many types of documents, notably CSS but also blog feeds, help documents, licenses, etc., to an HTML file

href: The CSS file's location is indicated by the href attribute. The CSS file's path must lead to the appropriate folder.

rel: The item's status as a stylesheet is indicated by the rel attribute.

4.4 Structure of CSS

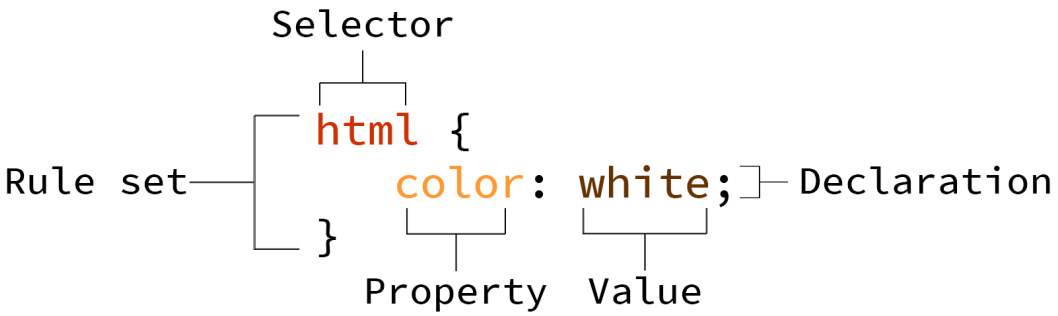

Figure 4.2: Structure of CSS

Despite having extremely basic components, CSS allows for sophisticated designs.

1. **rule set:** a collection of CSS designs for a specific set of elements
2. **declaration:** a single CSS line that adds a design
3. **selector:** the target item on which the design property is to be applied
4. **property:** the type of design to be added
5. **value:** instructions/values for changing the design property

4.5 Targeting things in CSS

To apply style to an element in CSS, first the element must be targeted by using selector. The following are the different types of selectors:

1. **Tag:** Selects all the tags of the same type. Example:
   ```
   html {}
   h1 {}
   p {}
   ```
2. **Class (.):** Select an element with the matching class. Example:
   ```
   .nav {}
   .contact {}
   .masthead {}
   ```

3. **ID (#):** Select an element with the matching ID. Example:
```
#profile {}
#nav {}
#social {}
```

4. **Space (Descendant):** Select an element that's a descendant of another element. Example:
```
ul li {}
nav a {}
```

5. **Child (>):** Select an element directly inside another element. Example:
```
ul > li {}
h1 > span {}
footer > .social {}
```

6. **Adjacent sibling (+):** Select an element directly beside another element. Example:
```
h1 + p {}
hr + p {}
li + li {}
```

7. **General sibling (~):** Select an element that's at the same level. Example:
```
p ~ p {}
h1 ~ p {}
dd ~ dt {}
```

8. **Attribute ([]):** Select an element by its attribute. Mostly used for styling external links differently. Example:
```
[data-state="active"] {}
[href^= "http"] {}
[download] {}
```

4.6 Some basic properties of CSS

- color
- background-color
- background-image
- display (inline, block, inline-block, flexbox)
- height
- width
- margin
- padding
- border

- border-color
- border-width
- border-style
- border-radius
- font
- font-family
- font-style
- font-weight
- position (absolute, relative, fixed)
- z-index
- float
- clear
- overflow
- text-align
- box-sizing
- grid
- word-wrap
- word-spacing

CHAPTER 5: JAVASCRIPT

5.1 Introduction to JavaScript

JavaScript (JS) is a programming language that allows the development of dynamic web pages by utilizing dynamic client-side scripting. JavaScript is a programming language that enables the implementation of complex functions on web pages. It integrates features like providing interactive content, animated 2D/3D graphics, displaying timely content updates and more with a web page.

JavaScript forms the foundation of front-end web development and the backbone of the World Wide Web alongside HTML and CSS. As of 2022, approximately 98% of websites utilize JavaScript on the client side for webpage behavior, frequently integrating third-party libraries. A dedicated JavaScript engine is available in every major web browser and is used to run the code on users' devices.

5.2 JavaScript Language Basics

The block of JavaScript code is generally executed in order, from top to bottom. Through the Document Object Model (DOM) API, JavaScript is frequently used to dynamically edit HTML and CSS to update a user experience. Web documents' code typically loads and runs in the order it appears on the page. If JavaScript is loaded and run before the HTML and CSS that it is meant to modify, errors could occur.

JavaScript is a lightweight interpreted programming language that can add interactivity to a website. The JavaScript code is sent to the web browser in text for, where it is then executed. Technically speaking, the majority of contemporary JavaScript interpreters actually employ a process called "just-in-time compiling" to enhance performance. This method converts JavaScript source code into a quicker, binary format while the script is being used, allowing it to run as quickly as possible. Since the compilation is handled during executed rather than beforehand, JavaScript is still regarded as an interpreted language.

Adding JavaScript to a webpage: Similar to how CSS is applied to an HTML page, JavaScript utilizes the same method. While CSS employs <link> elements to apply external stylesheets and <style> elements to apply internal stylesheets, JavaScript uses the <script> element.

This can be done using the following ways:

- **Internal JavaScript:** In this method, the JavaScript code is included internally within the HTML document itself. This code is written within the <script> tag which is enclosed by the <head> tag.

Example:

```
<head>
<script>
// JavaScript goes here
</script>
</head>
```

- **External JavaScript:** In this method, the JavaScript code is stored in a separate .js file which is then linked to the HTML document using the <script> tag within the <head> tag. Example:

```
<script src="script.js" defer></script>
```

Variables: They are the containers that store values. Declaration of a variable is done with the let keyword, followed by the name assigned to the variable. Example:

```
let myVariable;
```

After declaring a variable, it can be assigned a value like this:

```
myVariable = “Manvendra”;
```

Declaration and assignment can also be done on the same line like this:

```
let myVariable = “Manvendra”;
```

Data Types: JavaScript supports the following data types:

Variable	Explanation	Example
String	This is a sequence of text known as a string. To signify that the value is a string, enclose it in single quote marks.	let myVariable = 'Manvendra';
Number	This is a number. Numbers don't have quotes around them.	let myVariable = 10;
Boolean	This is a True/False value. The words true and false are special keywords that don't need quote marks.	let myVariable = true;
Array	This is a structure that allows you to store multiple values in a single reference.	let myVariable = [1,'Manvendra','Singh',10];
Object	This can be anything. Everything in JavaScript is an object and can be stored in a variable.	let myVariable = document.querySelector('h1');

Table 5.1: Data Types in JavaScript

Operators: An operator is a mathematical symbol that produces a result based on two values or variables. Some of the most common operators are shown below:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Assignment (=)
- Strict equality (===)
- Not (!)
- Does-not-equal (!==)

Functions: Functions in JavaScript is a block of code designed to perform a particular task. A JavaScript function is executed when something invokes it.

Syntax of a JavaScript function

- The function keyword is used to define a JavaScript function, which is then followed by the function's name and parenthesis ().
- Function names may also include underscores, dollar signs, and other characters (same rules as variables).
- Names of parameters may be included in parenthesis and separated by commas.
- Curly brackets enclose the code that the function will execute
- The function definition lists the function parameters between parentheses ().

```
function name(parameter1, parameter2, parameter3,…)
{
      // code to be executed
}
```

- The function definition lists the function parameters between parentheses ().
- When a function is called, values are passed to it as arguments.
- The arguments behave as local variables within the function.

Calling a function: Making use of a function requires calling it after it has been defined.
A function can be called by separating the function name from the argument values in parenthesis, followed by a semicolon.
The JavaScript function calling syntax is demonstrated below.

```
function name(value1, value2, value3,…);
```

5.3 JavaScript Loops, Condition

Loops: Loops allow the execution of the same code over and over again. Every time the loop is repeated, the code is frequently somewhat changed, or it runs with different variables.
JavaScript supports the following looping statements:

- for loop
- for/in loop
- for/of loop
- while loop
- do/while loop

Conditionals: Conditional statements allow the process of decision making in JavaScript code. It allows the execution of different actions for different decisions.
JavaScript supports the following conditional statements:

- if statement
- else statement
- else if statement
- switch statement

5.4 JavaScript Objects

As JavaScript is an Object-Oriented Programming Language, almost everything is an object in JavaScript. All JavaScript values, except primitives, are objects. A primitive value is a value that has no properties or methods.

Object Definition: JavaScript object can be defined and created with an object literal:
Example:

```
const person = {firstName: “Manvendra”, lastName: “Singh”, age: 25, branch: “CS”};
```

Object Properties: The named values, in JavaScript objects, are called properties.
Example:

Property	Value
firstName	Manvendra
lastName	Singh
age	25
branch	CS

Table 5.2: Object Properties in JavaScript Example

Object Methods: Methods are actions that can be performed on objects. Methods are stored in properties as function definitions. A method is a function stored as a property.

Property	Value
firstName	Manvendra
lastName	Singh
age	25
branch	CS
fullName	function() {return this.firstName + " " + this.lastName;}

Table 5.3: Object Methods in JavaScript Example

5.5 Basics of ES6

ECMAScript 2015 was the second major revision to JavaScript. It is also known as ES6 or ECMAScript 6.

The new features in ES6 are:

- The let and const keywords
- Arrow Functions
- The ... Operator
- For/of
- Map Objects
- Set Objects
- Classes
- Promises
- Symbol
- Default Parameters
- Function Rest Parameter
- String.includes()
- String.startsWith()
- String.endsWith()
- Array.from()
- Array keys()
- Array find()
- Array findIndex()
- New Math Methods
- New Number Properties
- New Number Methods
- New Global Methods
- Object entries
- JavaScript Modules

5.6 JavaScript DOM

The objects that make up a web document's structure and content are represented as data by the Document Object Model (DOM).

The Document Object Model (DOM) is a programming interface for web documents. It serves as a representation of the page so that programs can alter the document structure, appearance and content. Programming languages like JavaScript can communicate with a page by interacting with the nodes and objects that the DOM uses to represent the document. DOM representation allows a web page to be manipulated. It is a representation of the web page that is object-oriented and can be changed using a scripting language like JavaScript.

It is called an Object Model because documents are modelled using objects and the model takes into account not only the structure of a document but also its behaviors and the objects that make up its constituent parts, such as the HTML tag elements and attributes.

The need for DOM: The web pages are organised using HTML, behaviour is added using JavaScript. The HTML document cannot be immediately understood by JavaScript when it is loaded into the browser. Consequently, a related document is produced (DOM). The DOM essentially represents the same HTML document in a new way by utilising objects. JavaScript can readily interpret DOM; for example, it can understand object h1 in DOM but not tags (<h1>H</h1>) in HTML documents. Now, JavaScript can use several functions to access each of the objects (h1, p, etc.).

Structure of DOM: The structure of DOM can be compared to that of a tree or forest (multiple trees). The tree-like representation of a document is also referred to as a "structural model". There are nodes at the ends of the tree's branches, and each node has objects in it. Nodes have the option of adding event listeners, which are activated when a specified event occurs. The ability to generate the same structure model with precisely the same objects and connections using any two DOM implementations is known as structural isomorphism, and it is a key characteristic of DOM structure models.

Methods of DOM:

- **write("string"):** Writes the given string on the document.
- **getElementById():** returns the element having the given id value.
- **getElementsByName():** returns all the elements having the given name value.
- **getElementsByTagName():** returns all the elements having the given tag name.
- **getElementsByClassName():** returns all the elements having the given class name.

CHAPTER 6: PROJECT

Using the various skills acquired by during the duration of my practical training in the various disciplines of front-end web development, I was able to create a clone of the popular web-word game called Wordle from scratch using HTML, CSS and JavaScript.

In this game, the player needs to guess a five-letter word in six attempts. The hints are provided after each attempt by the use of different colored tiles. A green tile indicates that the letter is in the correct position, a yellow tile indicates that the letter is in the final word but is not placed at the correct position and a gray tile indicates that the letter is not in the final word.

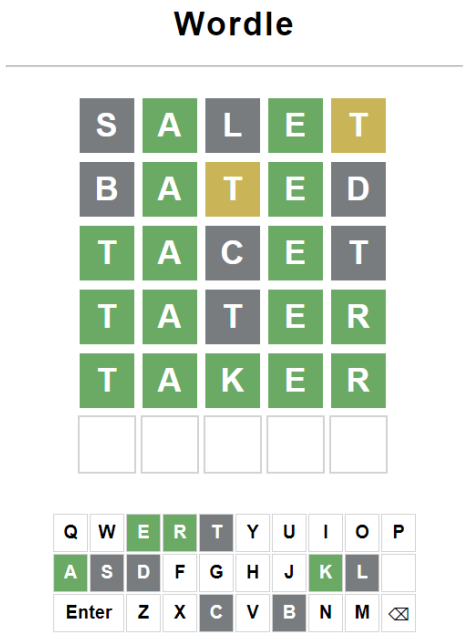

Figure 6.1: Wordle Project made using HTML, CSS and JavaScript

This project uses the programming logic of JavaScript to randomly select a word from a long list of five-letter words each time the HTML page is reloaded. The comparison of the input to the correct word is also handled by JavaScript. On each attempt, proper feedback is provided to the player in the form of colored tiles. For this project, CSS is utilized to create an aesthetically pleasing interface while the HTML code is minimal as most of the elements are generated by JavaScript.

CONCLUSION

Summing up this whole report, this practical training was an educational and rewarding experience. Needless to say, the skills I acquired during the duration of my training will prove to be highly useful in my career as an IT professional. The experience I received during the span of 45 days while working and coordinating with fellow professionals also improved my soft-skills like teamwork, communication and management. Although I had prior experience in the field of web development, working in a professional organization helped me gain first-hand experience of the industry while allowing me to explore the practical applications of my technical knowledge which is not possible in a traditional classroom.

The project I developed during the practical training also improved my knowledge of HTML, CSS and JavaScript as I was able to utilize various aspects of programming logic in my Wordle clone. It broadened my horizon of thinking and will surely prove useful in the time to come.

In the future, I will keep improving my web-development skills while also acquiring the knowledge of other disciplines like DSA which will help me integrate the knowledge of various fields to create a better project.

REFERENCES

[1] Martin, Sarah. HTML, CSS, and JavaScript. The Definitive Guide to Squarespace, 2014.

[2] Duckett, Jon. Web design with HTML, CSS, JavaScript and jQuery set. Vol. 1. IN: Wiley, 2014.

[3] Goodman, Danny. Dynamic HTML: The definitive reference: A comprehensive resource for HTML, CSS, DOM & JavaScript. "O'Reilly Media, Inc.", 2002.

[4] https://www.w3schools.com/tags/ref_byfunc.asp

[5] Duckett, Jon. HTML & CSS: design and build websites. Vol. 15. Indianapolis, IN: Wiley, 2011