Competency Based Learning Materials (CBLMs)

Web Design and Development for Freelancing

Level-3

Module 5: Develop website using JavaScript

Code: CBLM-ICT-WDF-05-L3-EN-V1

National Skills Development Authority
Prime Minister's Office
Government of the People's Republic of Bangladesh

Copyright

National Skills Development Authority
Prime Minister's Office
Level: 10-11, Biniyog Bhaban,
E-6 / B, Agargaon, Sher-E-Bangla Nagar Dhaka-1207, Bangladesh.
Email: ec@nsda.gov.bd
Website: www.nsda.gov.bd.
National Skills Portal: http:\\skillsportal.gov.bd

The CBLM on "Develop website using JavaScript" is developed based on NSDA approved Competency Standards and Competency Based Curriculum under Web Design and Development for Freelancing Level-3 Occupation. It contains the information required to implement the Web Design and Development for Freelancing Level-3 standard.

This document has been prepared by NSDA with the help of relevant experts, trainers/professionals.

All Government-Private-NGO training institutes in the country accredited by NSDA can use this CBLM to implement skill-based training of Web Design and Development for Freelancing Level-3 course.

How to use this Competency Based Learning Materials (CBLMs)

The module, Maintaining and enhancing professional & technical competency contains training materials and activities for you to complete. These activities may be completed as part of structured classroom activities or you may be required you to work at your own pace. These activities will ask you to complete associated learning and practice activities in order to gain knowledge and skills you need to achieve the learning outcomes.

1. Review the **Learning Activity** page to understand the sequence of learning activities you will undergo. This page will serve as your road map towards the achievement of competence.

2. Read the **Information Sheets.** This will give you an understanding of the jobs or tasks you are going to learn how to do. Once you have finished reading the **Information Sheets** complete the questions in the **Self-Check.**

3. **Self-**Checks are found after each **Information Sheet**. **Self-Checks** are designed to help you know how you are progressing. If you are unable to answer the questions in the **Self-Check** you will need to re-read the relevant **Information Sheet**. Once you have completed all the questions check your answers by reading the relevant **Answer Keys** found at the end of this module.

4. Next move on to the **Job Sheets. Job Sheets** provide detailed information about *how to do the job* you are being trained in. Some **Job Sheets** will also have a series of **Activity Sheets**. These sheets have been designed to introduce you to the job step by step. This is where you will apply the new knowledge you gained by reading the Information Sheets. This is your opportunity to practise the job. You may need to practise the job or activity several times before you become competent.

5. Specification **sheets**, specifying the details of the job to be performed will be provided where appropriate.

6. A review of competency is provided on the last page to help remind if all the required assessment criteria have been met. This record is for your own information and guidance and is not an official record of competency

When working though this Module always be aware of your safety and the safety of others in the training room. Should you require assistance or clarification please consult your trainer or facilitator.

When you have satisfactorily completed all the Jobs and/or Activities outlined in this module, an assessment event will be scheduled to assess if you have achieved competency in the specified learning outcomes. You will then be ready to move onto the next Unit of Competency or Module

Approved by

---th Executive Committee (EC) Meeting of NSDA

Held on ----------

Table of Contents

Module Content

Module: Develop Website Using JavaScript

Module Descriptor: This module encompasses the necessary knowledge, skills, and attitudes (KAS) for establishing work to develop a website using JavaScript. It specifically includes planning for the website, developing it using JavaScript and testing it.

Nominal Hours: 80

Learning Outcome 1: After completion of the module, trainees will be able to:

1. Plan for website
2. Develop a website using JavaScript.
3. Test the website.

Learning Outcomes:

By the end of this lesson, learners will be able to:

Assessment Criteria:

1.1 The purpose and intended audience of the website are identified.
1.2 Functional requirements and constraints are identified.
1.3 A coding plan is developed as required.
1.4 The necessary software is installed, and functionality is checked.

2.1 JavaScript is coded as per functional requirements.
2.2 JavaScript files are integrated with website as per standard procedure.
2.3 JavaScript is executed to finalize the website.

3.1 The website is tested to ensure functionality and errors are corrected as per standard operating procedure.
3.2 The website is opened with common browsers and check for accessibility, readability, legibility and presentation in accordance with client requirements.
3.3 The website is evaluated for fitness in terms of the purpose, target audience and specifications of client requirements.

Learning Outcome 1: Plan a website

Assessment Criteria	1. The purpose and intended audience of the website are identified. 2. Functional requirements and constraints are identified. 3. A coding plan is developed as required. 4. Necessary software installed as per requirement.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standards. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. Identifying the Purpose and Intended Audience of the Website 2. Identifying Functional Requirements and Constraints 3. Developing a Coding Plan as Needed 4. Installing Necessary Software as per Requirements
Training Methods	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience 1: Plan a website.

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Students will ask the instructor about Independently setting up client servers.	1. The instructor will provide the learning materials for` Independently setting up client servers.
2. Read the **Information sheet/s**	2. Information Sheet No 1: Plan a website. ▪ Identifying the Purpose and Intended Audience of the Website ▪ Identifying Functional Requirements and Constraints ▪ Developing a Coding Plan as Needed ▪ Installing Necessary Software as per Requirements
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No 1: Plan a website. Answer key No 1: Plan a website.
1. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Job Sheet No 1: Plan a website. Specification Sheet 1: Plan a website.

Information Sheet 1 Plan a website

Learning Objective: After completing this information sheet, the learners will be able to Plan a website.

1.1 Purpose and Intended Audience of the Website
1.2 Functional Requirements and Constraints
1.3 Coding Plan as Needed
1.4 Necessary Software as per Requirements

1.1 Purpose and Audience Identification:

Purpose of the Website

A website's purpose is its foundation, the reason it exists. Before diving into design and development, defining why you're creating a website is crucial. Is it for showcasing your portfolio, selling products, sharing information, or providing services? The purpose is to give your website direction. When embarking on the journey of creating a website, defining its purpose is your very first step. Think of this as the North Star guiding your entire web development process. Why does your website exist, and what role will it play in the digital realm? To aid self-learning, let's dive deeper into understanding this concept with examples and discussions.

Why Purpose Matters

- Clarity and Focus: A well-defined purpose ensures clarity. It's like setting a destination before starting a journey. It keeps you focused on what you want to achieve with your website.

  Examples:
  - Personal Portfolio: Imagine you're a photographer whose website aims to showcase your best work. In this case, every design choice, from layout to image quality, should revolve around presenting your portfolio effectively.
  - E-commerce: If you're running an online store, your website's purpose is clear: selling products. Everything, from product listings to checkout, should be optimized for this purpose.

- Audience Alignment: Knowing your purpose helps you connect with the right audience. Different purposes attract different visitors.

  Examples:
  - Educational Platform: If your website aims to provide educational content, it's more likely to attract students and educators interested in learning and teaching.
  - Entertainment Blog: If your purpose is to entertain through blogs or videos, your audience will be people seeking entertainment or information on specific topics.

How to Define Your Website's Purpose

- Identify Your Goals: Ask yourself what you want to achieve with your website. Are you looking to inform, entertain, sell, or connect? Your goals are closely tied to your website's purpose.
- Understand Your Audience: Consider who your target audience is. Understanding their needs and preferences helps align your purpose with what your audience seeks.

Identifying the Intended Audience

Understanding your audience is fundamental. Are you targeting consumers, businesses, students, hobbyists, or a specific niche? Knowing your audience helps tailor content, design, and functionality to meet their needs and preferences.

1.2 Identifying Functional Requirements and Constraints

Before developing a website using JavaScript, it's crucial to identify and define the functional requirements and constraints that will shape your project. This information sheet outlines the importance of this step and provides guidance on effectively identifying and documenting functional requirements and constraints for your web development project.

Why Identify Functional Requirements and Constraints?

Identifying functional requirements and constraints is a critical initial phase of any web development project. It serves several essential purposes:

- Clarity of Objectives: It helps clarify the goals and objectives of the website. Understanding what the website needs to achieve is fundamental to its success.
- Scope Definition: It defines the project's scope, ensuring that everyone involved understands what will and won't be included in the website.
- Resource Allocation: It assists in allocating the necessary resources, such as time, budget, and human resources, effectively.
- Risk Management: Identifying constraints and potential challenges early allows for proactive risk management.

Identifying Functional Requirements:

Functional requirements describe the specific features and functionalities the website must have to meet its objectives. Here's how to identify them:

- Stakeholder Interviews: Engage with stakeholders, including clients, users, and project managers, through interviews or meetings. Ask them about their expectations and what they envision the website doing.

- User Stories: Create user stories to capture how different types of users will interact with the website. User stories help in understanding the user's perspective and needs.

- Use Cases: Develop use cases that outline different scenarios in which the website will be used. These can help identify specific functionalities required for each scenario.

- Functional Prototypes: Create functional prototypes or wireframes to visualise the website's layout and features. This often leads to a clearer understanding of what is needed.

- Feedback from Similar Projects: Gather insights and feedback from similar projects or websites if applicable. This can provide valuable insights into what works and what doesn't.

Identifying Constraints:

Constraints are limitations or conditions that may impact the development of the website. They need to be identified and managed effectively. Here's how to do it:

- Technical Constraints: Understand the technical limitations, including the hosting environment, compatibility issues, and any specific technologies or platforms that must be used.
- Budget and Time Constraints: Determine the budget and timeline for the project. These constraints will influence decisions regarding the scale and complexity of the website.
- Legal and Compliance Constraints: Identify any legal requirements or industry-specific compliance standards the website must adhere to, such as data protection regulations.
- Resource Constraints: Assess the availability of resources, including human resources, tools, and software licenses.
- Security Constraints: Identify security requirements and constraints. This is especially important if the website handles sensitive data.

1.3 Developing a Coding Plan as Needed

JavaScript is a powerful programming language for enhancing website interactivity and functionality. To build a successful website using JavaScript, developing a coding plan that outlines how you'll structure, write, and manage your JavaScript code is essential.

Why Develop a Coding Plan?

Developing a coding plan is a crucial step in web development with JavaScript. It serves several important purposes:

- Structure and Organization: A coding plan helps you structure your code, ensuring it's organised, modular, and easy to maintain.

- Efficiency: It helps you work more efficiently by providing a roadmap for coding tasks and objectives.
- Collaboration: If you're working in a team, a coding plan ensures that everyone understands the code's structure and logic, facilitating collaboration.
- Scalability: A well-thought-out plan makes your codebase scalable, allowing you to add new features or make changes without introducing errors.
- Debugging and Maintenance: It simplifies debugging and maintenance by providing a clear reference for code components.

Steps to Develop a Coding Plan:

- Project Understanding: Begin by thoroughly understanding the project's requirements and objectives. This includes identifying the specific functionalities and interactions your JavaScript code will need to support.

- Breakdown of Tasks: Divide your project into smaller, manageable tasks or features. Each task should represent a specific functionality you'll implement with JavaScript.

- Flowcharts and Diagrams: Consider using flowcharts or diagrams to visualise the flow of your JavaScript code. This can help in understanding how different components will interact.

- Pseudocode: Write pseudocode for complex functionalities or algorithms before diving into coding. Pseudocode is a human-readable way to outline the logic of your code.

- Modularisation: Plan how to break down your code into modules or functions. This promotes code reusability and maintainability.

- File Structure: Decide on the file structure for your JavaScript code. Organise your files logically and establish naming conventions for consistency.

- Coding Standards: Establish coding standards and guidelines for your project. This includes naming conventions, indentation style, and commenting practices.

- Testing and Debugging: Define your testing and debugging approach. Plan how you'll test each feature or functionality as you build it and how you'll troubleshoot and fix issues.

- Version Control: If you work in a team, decide on version control practices and tools (e.g., Git) to manage code changes.

- Documentation: Include documentation in your coding plan. Document your code, including function descriptions and usage, to make it easier for you and others to understand.

- Error Handling: Plan handling errors and exceptions in your JavaScript code. Consider using try-catch blocks or other error-handling mechanisms.

- Performance Optimization: If your project requires high performance, outline how you'll optimise your JavaScript code for speed and efficiency.

- Security Considerations: Identify security considerations, such as data validation and protection against common web vulnerabilities (e.g., Cross-Site Scripting or XSS).

1.4 Installing Necessary Software

Identify the software tools required for website design and development. Common choices include web design software like Adobe XD, Sketch, or Figma and development tools like code editors (e.g., Visual Studio Code, Atom, Bracket, Sublime Text, Text pad, and Notepad++).

Install Visual Studio

Step 1. Make sure your computer is ready for Visual Studio.

Before you begin installing Visual Studio:

- Check the system requirements
  - https://learn.microsoft.com/en-us/visualstudio/releases/2022/system-requirements. These requirements help you know whether your computer supports Visual Studio 2022.
- Ensure that the user performing the installation has administrator permissions on the machine.
- Apply the latest Windows updates. These updates ensure that your computer has the latest security updates and the required system components for Visual Studio.
- Reboot. The reboot ensures that pending installs or updates don't hinder your Visual Studio install.
- Free up space. Remove unneeded files and applications from your system drive by, for example, running the Disk Cleanup app.

Step 2. Initiate the installation.

The latest release of Visual Studio 2022 is hosted on Microsoft servers. To install this, click the following link and choose your desired edition. A small "bootstrapper" file will be downloaded into your Downloads folder.

- https://visualstudio.microsoft.com/downloads/?cid=learn-onpage-download-cta

- Double-click the bootstrapper from your Downloads folder, VisualStudioSetup.exe, or name something like vs_community.exe to start the installation.

- If you receive a User Account Control notice, choose Yes. A window will ask you to acknowledge the Microsoft License Terms and the Microsoft Privacy Statement. Choose Continue.

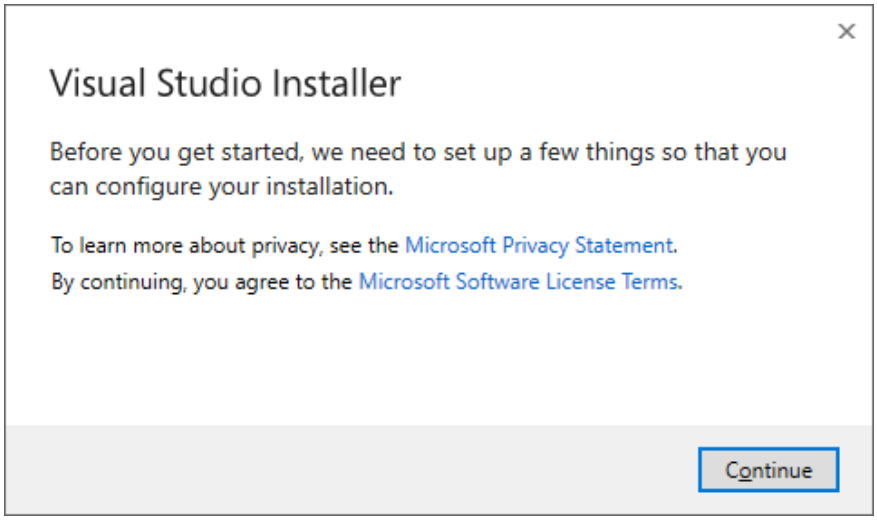


Figure 1: Video Studio Installer.

Step 3. Choose workloads.

After the Visual Studio Installer is installed, you can use it to customize your installation by selecting the feature sets—or workloads—that you want. After you choose the workload(s) you want, select **Install**.

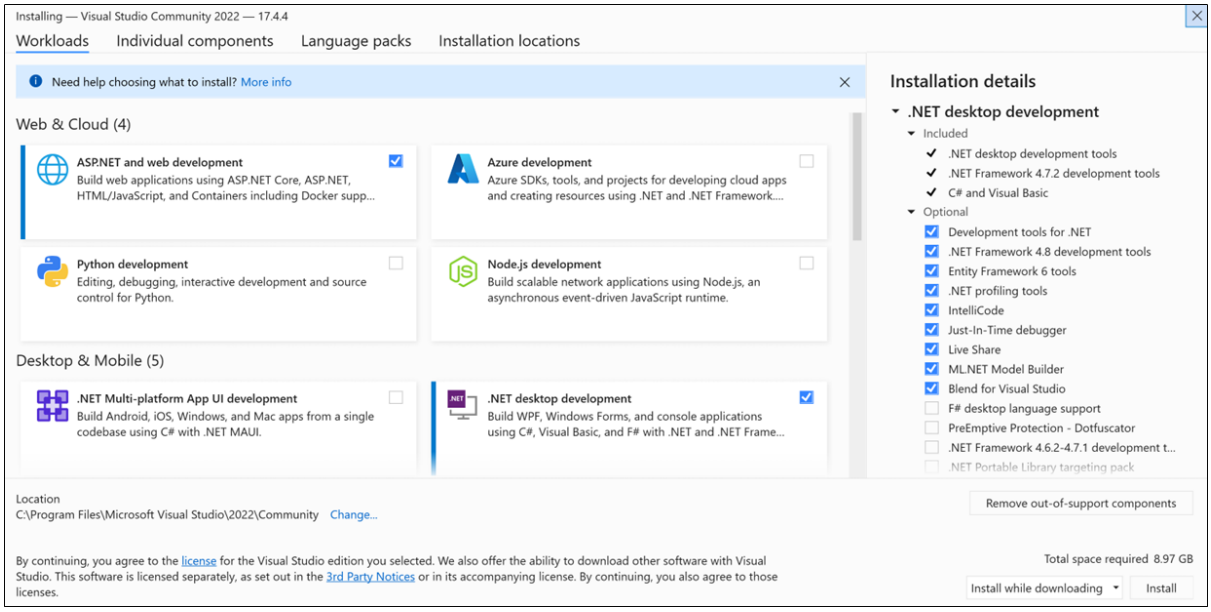


Figure 2: Choose Workloads in Visual Studio.

Step 4. Choose individual components (optional)

If you don't want to use the Workloads feature to customize your Visual Studio installation, or you want to add more components than a workload installs, you can do so by installing or adding individual components from the **Individual components** tab. Choose what you want, and then follow the prompts.

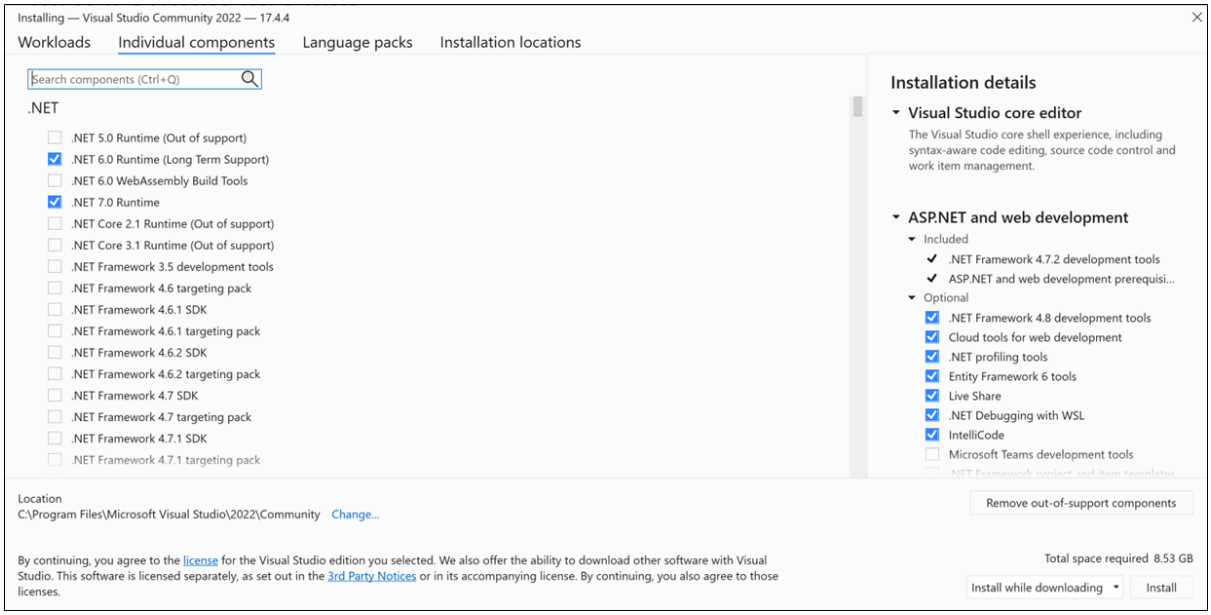


Figure 3: Choose Individual Component in VS Studio.

Step 5. Install language packs (optional)

By default, the installer program tries to match the operating system's language when it runs for the first time. To install Visual Studio in your chosen language, choose the **Language Packs** tab from the Visual Studio Installer and then follow the prompts.

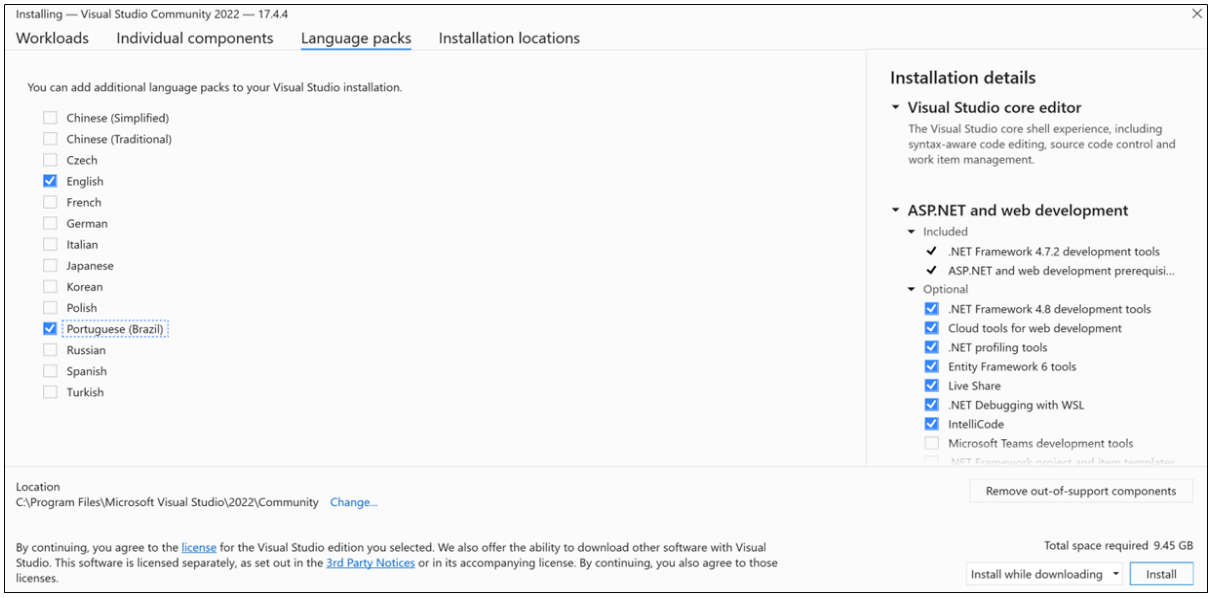


Figure 4: Install language packs in Visual Studio.

Step 6. Select the installation location (optional)

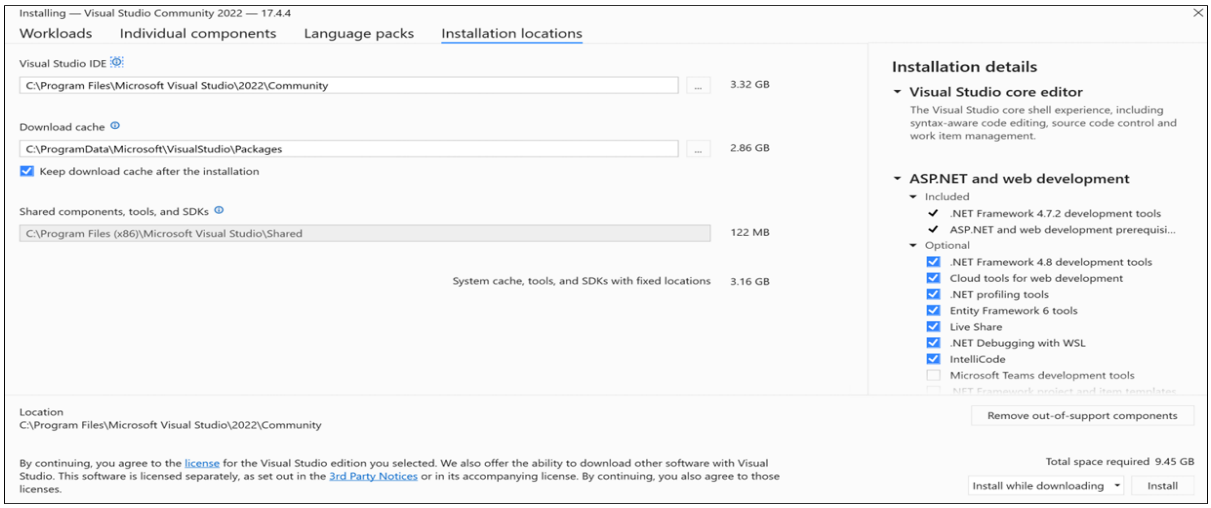


Figure 5: Select the installation location of Visual Studio.

You can select a different drive for Visual Studio IDE or Download cache only when you first install Visual Studio. If you've installed Visual Studio on your computer before, you won't be able to change the Shared components, tools, and SDKs path, and it will appear greyed out. All installations of Visual Studio share this location.

Step 7. Start developing.

- After completing your Visual Studio installation, select the Launch button to start developing with Visual Studio.
- On the start window, choose Create a new project.
- In the template search box, enter the type of app you want to create to see a list of available templates. The list of templates depends on the workloads that you choose during installation. To see different templates, choose different workloads.
  You can also filter your search for a specific programming language by using the Language drop-down list. You can filter by using the Platform list and the Project type list, too.
- Visual Studio opens your new project, and you're ready to code!

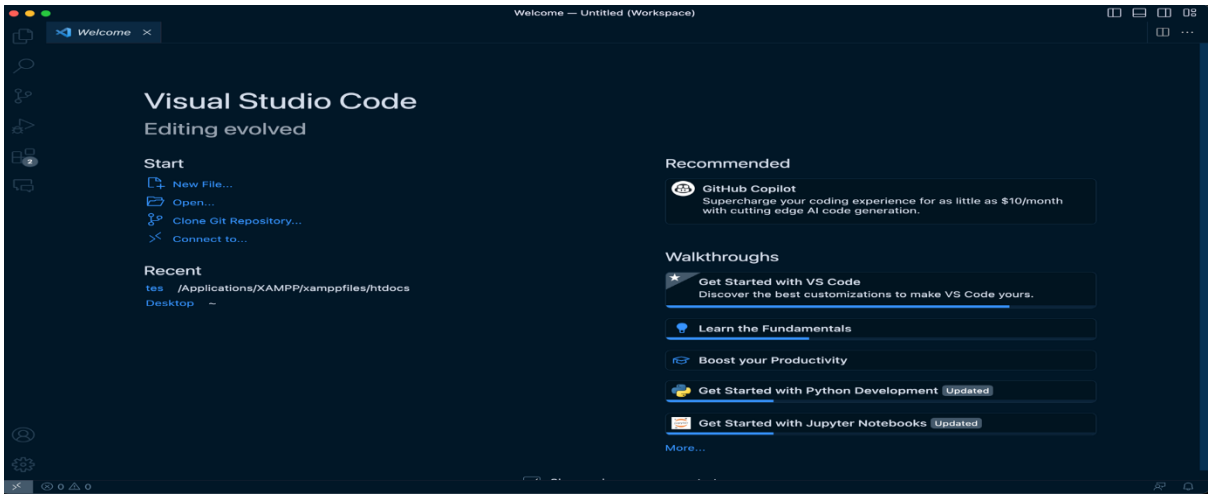


Figure 6: Visual Studio.

Self-Check Sheet 1 Plan a website

1. What is the significance of defining the purpose of a website in the web development process?

    a) It determines the color scheme of the website.
    b) It aligns the website's design with current trends.
    c) It provides clarity and focus for the project.
    d) It impacts the website's loading speed.

2. Why is it essential to identify the intended audience of a website?

    a) To determine the website's purpose.
    b) To decide on the coding plan.
    c) To align the design with current trends.
    d) To choose the color scheme.

3. What is the primary purpose of functional requirements in web development?

    a) To define the project's budget.
    b) To specify coding standards.
    c) To describe specific features and functionalities.
    d) To outline the website's color scheme.

4. Why is it crucial to identify constraints in a web development project?

    a) Constraints ensure unlimited resources for the project.
    b) Constraints help allocate resources efficiently.
    c) Constraints prevent any changes to the project scope.
    d) Constraints are not relevant in web development.

5. What is the purpose of a coding plan in web development with JavaScript?

    a) It defines the website's purpose.
    b) It specifies the color scheme.
    c) It provides a roadmap for coding tasks.
    d) It determines the project's budget.

6. How can modularization be beneficial in coding with JavaScript?

    a) It adds complexity to the code.
    b) It makes the code less reusable.
    c) It increases the file size.
    d) It promotes code reusability and maintainability.

7. What should a coding plan include to help with debugging and maintenance?

    a) A list of all team members.
    b) A detailed project timeline.
    c) Function descriptions and usage documentation.

d) The project's color scheme.

8. Why is version control important in web development, especially for team projects?

a) It determines the project's budget.
b) It prevents any code changes.
c) It helps manage code changes and collaboration among team members.
d) It defines the project's color scheme.

9. What is the role of pseudocode in the coding planning process?

a) It defines the website's purpose.
b) It serves as a human-readable outline of code logic.
c) It specifies coding standards.
d) It determines the project's budget.

10. Why should security considerations be part of the coding plan in web development?

a) Security is not relevant in web development.
b) Security is the sole responsibility of the hosting provider.
c) Security ensures that sensitive data is protected.
d) Security only impacts the website's colour scheme.

Answer Sheet 1 Plan a website

1. c) It provides clarity and focus for the project.
2. a) To determine the website's purpose.
3. c) To describe specific features and functionalities.
4. b) Constraints help allocate resources efficiently.
5. c) It provides a roadmap for coding tasks.
6. d) It promotes code reusability and maintainability.
7. c) Function descriptions and usage documentation.
8. c) It helps manage code changes and collaboration among team members.
9. b) It serves as a human-readable outline of code logic.
10. c) Security ensures that sensitive data is protected.

Job Sheet 1 IDE Installation for CSS Coding

Job Sheet Title: IDE Installation for CSS Coding

Objective: Install and set up an Integrated Development Environment (IDE).

Tasks:

Step 1. Choose an IDE

- Research and select an IDE suitable for CSS coding. Popular choices include Visual Studio Code, Sublime Text, and Atom.
- Consider factors like user-friendliness, extensions/plugins available, and your operating system's compatibility.

Step 2. Download and Install the IDE

- Visit the official website of the selected IDE using a web browser.
- Locate the download section and download the installer for your operating system (e.g., Windows, macOS, Linux).
- Run the downloaded installer and follow the installation instructions.

Step 3. Basic Setup

- Launch the IDE after installation.
- Customise the IDE settings according to your preferences (e.g., theme, font size, indentation).
- Familiarise yourself with the IDE's user interface.

Step 4. Install CSS Extensions/Plugins

- Most modern IDEs support extensions or plugins to enhance functionality.
- Search for and install CSS-related extensions or plugins. These may include linters, autocompletion, and colour pickers.
- Configure the extensions as needed.

Step 5. Create a Sample CSS File

- Open your new IDE.
- Create a sample CSS file with a .css extension.
- Add some CSS rules to the file to test your environment.

Step 6. Test the Environment

- Write CSS code in your IDE.
- Save the CSS file.
- Open a web browser and create an HTML file.
- Link your HTML file to the CSS file you created.
- View the webpage in the browser to ensure your CSS is applied correctly.

Specification Sheet 1 IDE Installation for CSS Coding

Job Title: IDE Installation for CSS Coding

Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer/Laptop	Minimum Corei3 with 4GB RAM	No.	1
3	Software (Browser)	Latest Version	No.	01
4	Internet connections	High Speed	No.	01

Other Specifications:

1. Chosen IDE for CSS - Visual Studio Code
2. Latest version available at the time of installation
3. Operating System: Windows 10
4. Customization:
   - Theme: Dark+
   - Font Size: 16px
   - Tab Width: 4 spaces
   - Plugins/Extensions Installed:
   - "Close Tag" for HTML tag completion
5. Test Procedure:
   1. Launch Visual Studio Code.
   2. Open a sample CSS file.
   3. Write and save CSS code.
   4. Create an HTML file.
   5. Link the HTML file to the CSS file.
   6. Verify that CSS styles are correctly applied to the HTML elements.

Learning Outcome 2: Develop a website using JavaScript

Contents:

1. Coding JavaScript According to Functional Requirements.
2. Integrating JavaScript Files into Your Website: Best Practices.
3. Executing JavaScript for Website Finalization.

Assessment Criteria:

2.1 JavaScript is coded as per functional requirements.
2.2 JavaScript files are integrated with the website as per standard procedure.
2.3 JavaScript is executed to finalise the website.

Conditions:

Students/trainees must be provided with the following:

- Applicable tools, utensils, and equipment as prescribed by competency standards.
- Supply materials
- Relevant ingredients
- CBLM related to the learning outcome.
- Instructions, job sheets, activity sheets, and standard operating procedures
- Personal protective equipment
- Module/reference

Learning Materials:

- CBLM
- Handouts
- Books, Manuals
- Module/ Reference
- Paper
- Pen

Learning Experience 2: Develop a website using JavaScript

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Students will ask the instructor about develop website using JavaScript.	1. The instructor will provide the learning materials for develop a website using JavaScript.
2. Read the **Information sheet/s**	2. Information Sheet No 2: Develop Website Using JavaScript. ▪ Coding JavaScript According to Functional Requirements. ▪ Integrating JavaScript Files into Your Website: Best Practices. ▪ Executing JavaScript for Website Finalization.
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No: Develop Website Using JavaScript. Answer key No. Develop Website Using JavaScript
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Job Sheet No: Specification Sheet:

Information Sheet 2 Develop a website using JavaScript

Learning Objective: After completing this information sheet, the learners will be able to develop a website using JavaScript.

2.1 JavaScript Core Components

a. Variables:

- Variables in JavaScript are used to store and manipulate data.
- They can hold various types of data, such as numbers, strings, booleans, objects, and more.
- Variables are declared using the var, let, or const keyword, followed by the variable name.

Example:

```javascript
var age = 25;
let name = "John";
const PI = 3.14;
```

b. Functions:

- Functions in JavaScript are reusable blocks of code that perform a specific task.
- They can accept input parameters (arguments) and return a value.
- Functions are defined using the function keyword, followed by the function name and a pair of parentheses.

Example:

```javascript
function sayHello() {
  console.log("Hello!");
}

function addNumbers(a, b) {
  return a + b;
}
```

c. Loops:

- Loops in JavaScript allow you to repeatedly execute a block of code.
- The for, while, and do-while loops are commonly used in JavaScript.
- Loops help automate repetitive tasks or iterate over collections of data.

Example:

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

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

d. Conditions:

- Conditions in JavaScript are used to perform different actions based on different conditions.
- The if, else if, and else statements are used for conditional branching.
- Conditions help control the flow of code execution.

Example:

```javascript
let age = 18;

if (age >= 18) {
  console.log("You are an adult.");
} else {
  console.log("You are a minor.");
}
```

e. Switches:

- The switch statement in JavaScript is used to perform different actions based on different cases.
- It provides an alternative to multiple if statements.

Example:

```javascript
let day = "Monday";

switch (day) {
  case "Monday":
    console.log("Today is Monday.");
    break;
  case "Tuesday":
    console.log("Today is Tuesday.");
    break;
  default:
    console.log("It's another day of the week.");
}
```

f. Objects:

- Objects in JavaScript are used to store multiple values as properties and methods.
- They can represent real-world entities and their attributes.
- Objects are created using the object literal {} or the new keyword.

Example:

```javascript
let person = {
  name: "John",
  age: 25,
  sayHello: function() {
    console.log("Hello, I'm " + this.name + ".");
  }
};
```

g. Arrays:

- Arrays in JavaScript are used to store multiple values in a single variable.
- They are ordered lists of values, indexed by numeric positions.
- Arrays are created using square brackets

Example:

```javascript
let numbers = [1, 2, 3, 4, 5];
let fruits = ["apple", "banana", "orange"];
```

h. Output:

- Output in JavaScript can be achieved using the **console.log()** function.
- It allows you to display information in the browser's console or the developer console.

Example:

```javascript
console.log("Hello, World!");
```

i. Comments:

- Comments in JavaScript are used to add explanatory notes to the code.
- They are ignored by the JavaScript engine and are not executed.
- Comments can be single-line (//) or multi-line (/* */).

Example:

```javascript
// This is a single-line comment.

/*
This is a multi-line comment.
It can span multiple lines.
*/
```

j. Data Types:

- JavaScript has several data types, including numbers, strings, booleans, objects, arrays, and more.
- Each data type represents a different kind of value and has different properties and behaviors.

Example:

```javascript
let age = 25; // Number
let name = "John"; // String
let isAdult = true; // Boolean
let person = { name: "John", age: 25 }; // Object
let numbers = [1, 2, 3, 4, 5]; // Array
```

k. Operators:

- Operators in JavaScript are used to perform operations on values.
- They include arithmetic operators, assignment operators, comparison operators, logical operators, and more.

Example:

```javascript
let a = 5;
let b = 3;

let sum = a + b; // Addition
let product = a * b; // Multiplication
let isGreater = a > b; // Comparison
let result = (a + b) * 2; // Parentheses for grouping
```

l. Comparisons:

- Comparison operators in JavaScript are used to compare values and return a boolean result.
- They are often used in conditional statements to make decisions.

Example:

```javascript
let a = 5;
let b = 3;

console.log(a > b); // true
console.log(a === b); // false
console.log(a !== b); // true
```

m. Breaks:

- The break statement in JavaScript is used to exit a loop or switch statement.
- It terminates the current loop iteration or case execution.

Example:

```javascript
for (let i = 0; i < 5; i++) {
  if (i === 3) {
    break; // Exit the loop when i is 3
  }
  console.log(i);
}
```

n. Errors:

- Errors in JavaScript can occur when there are syntax mistakes or runtime issues.
- JavaScript provides different types of errors, such as SyntaxError, ReferenceError, TypeError, etc.
- Errors can be caught and handled using try...catch blocks.

Example:

```javascript
try {
  // Code that may cause an error
  let x = y + 5; // ReferenceError: y is not defined
} catch (error) {
  // Handle the error
  console.log("An error occurred:", error.message);
}
```

p. Validation:

- Validation in JavaScript refers to the process of checking user input or data to ensure it meets certain criteria.
- It is commonly used in web forms to validate user-submitted data.
- Validation can involve checking for required fields, data formats, length constraints, and more.

Example:

```javascript
function validateForm() {
  let name = document.getElementById("name").value;
  if (name === "") {
    alert("Please enter your name.");
    return false;
  }
  // Other validation checks...
  return true;
}
```

Understanding the Basic JavaScript Platform

a. Node.js:

- Node.js is a JavaScript runtime environment that allows you to run JavaScript on the server-side.
- It provides a powerful platform for building scalable and high-performance web applications.
- Node.js uses an event-driven, non-blocking I/O model, making it efficient for handling concurrent requests.

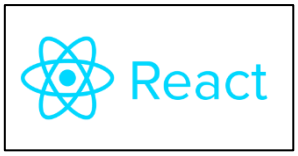
Example:

```javascript
// A simple Node.js server
const http = require('http');

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, Node.js!');
});

server.listen(3000, 'localhost', () => {
  console.log('Server running at http://localhost:3000/');
});
```

b. React.js:



- React.js is a JavaScript library for building user interfaces.
- It allows you to create reusable UI components and efficiently update the user interface based on changing data.
- React.js follows a component-based architecture, making it easy to build complex UIs.

Example:

```javascript
// A simple React component
import React from 'react';

class Hello extends React.Component {
  render() {
    return <h1>Hello, React.js!</h1>;
  }
}

export default Hello;
```

c. Vue.js:

- Vue.js is a progressive JavaScript framework for building user interfaces.
- It provides a simple and flexible approach to building interactive web applications.
- Vue.js can be used for small components or for developing large-scale applications.

Example:

```javascript
// A simple Vue.js component
<template>
  <h1>Hello, Vue.js!</h1>
</template>

<script>
export default {
  name: 'Hello',
}
</script>
```

d. Angular.js:

- Angular.js (also known as AngularJS) is a JavaScript framework for building web applications.
- It provides a comprehensive set of features for building dynamic and data-driven applications.
- Angular.js uses a two-way data binding approach, making it easy to synchronize data between the model and view.

Example:

```javascript
// A simple Angular.js component
angular.module('myApp', [])
  .controller('HelloController', function($scope) {
    $scope.message = 'Hello, Angular.js!';
  });
```

Note: The examples provided above are simplified representations of the respective platforms/frameworks. Each platform/framework has its own extensive documentation and features that you can explore for more in-depth understanding and usage.

Interpreting JavaScript Libraries

jQuery:

- jQuery is a fast and lightweight JavaScript library that simplifies HTML document traversal, event handling, and animation.
- It provides a wide range of utility functions and methods that make interacting with the Document Object Model (DOM) and manipulating web page elements easier.

Example:

```javascript
// Using jQuery to hide an element on button click
$(document).ready(function() {
  $('#myButton').click(function() {
    $('#myElement').hide();
  });
});
```

MooTools:

- MooTools is a compact JavaScript framework that provides a set of powerful tools for web development.
- It offers a collection of reusable and modular components for handling common tasks like DOM manipulation, Ajax requests, animations, and more.

Example:

```javascript
// Using MooTools to animate an element on hover
window.addEvent('domready', function() {
  $('myElement').addEvents({
    'mouseenter': function() {
      this.tween('background-color', '#ff0000');
    },
    'mouseleave': function() {
      this.tween('background-color', '#ffffff');
    }
  });
});
```

a. JS Charts:

- JS Charts is a JavaScript library that allows you to create interactive and customizable charts and graphs on web pages.
- It provides a range of chart types, including line charts, bar charts, pie charts, and more, with various customization options.

Example:

```html
<!-- Using JS Charts to create a bar chart -->
<!DOCTYPE html>
<html>
<head>
  <script src="jscharts.js"></script>
</head>
<body>
  <div id="chartContainer" style="width: 500px; height: 300px;"></div>

  <script>
    var myChart = new JSChart('chartContainer', 'bar');
    myChart.setDataArray([4, 9, 7, 5, 2, 6]);
    myChart.draw();
  </script>
</body>
</html>
```

b. ImageFX:

- ImageFX is a JavaScript library that provides image manipulation and effects capabilities.
- It allows you to apply various image filters, transformations, and effects to images on web pages.

Example:

```javascript
// Using ImageFX to apply a sepia effect to an image
var image = document.getElementById('myImage');
var fx = new ImageFX(image);
fx.sepia().apply();
```

c. Datejs:

- Datejs is a JavaScript library that simplifies working with dates and times.
- It provides a set of powerful and intuitive methods for parsing, manipulating, and formatting dates.

Example:

```javascript
// Using Datejs to calculate the number of days between two dates
var startDate = Date.parse('2023-01-01');
var endDate = Date.parse('2023-02-15');
var daysDifference = endDate.daysBetween(startDate);
console.log(daysDifference); // Output: 45
```

Note: The examples provided above are simplified representations of the respective libraries. Each library has its own extensive documentation and features that you can explore for more in-depth understanding and usage.

Interpreting BOM (Browser Object Model) and DOM (Document Object Model)

BOM (Browser Object Model):

- The Browser Object Model (BOM) represents the objects and interfaces web browsers provide to interact with the browser window.
- It provides methods and properties to control and manipulate the browser window, navigate through the browser history, and interact with browser-specific features.
- The BOM is not standardised, and different browsers may implement different features or have variations.

DOM (Document Object Model):

- The Document Object Model (DOM) represents the structure of an HTML or XML document as a tree-like structure.
- It provides a way to access and manipulate elements and content within a web page.
- The DOM treats an HTML or XML document as a collection of objects, where each element, attribute, and text node is represented as an object with properties and methods.
- The World Wide Web Consortium (W3C) standardises the DOM, ensuring a consistent interface across different browsers.

Interactions between BOM and DOM:

- The BOM and DOM work together to enable dynamic and interactive web page functionality.
- The BOM provides access to the browser window and its properties, such as the location, history, and screen dimensions.
- The DOM provides access to the structure and content of the web page, allowing manipulation of elements, attributes, and text nodes.
- JavaScript is commonly used to interact with both the BOM and DOM, as it provides methods and events to access and modify elements and properties within the browser window and web page.

Example:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Interacting with BOM and DOM</title>
  <script>
    function changeBackgroundColor() {
      // Accessing the DOM to change the background color of an element
      var element = document.getElementById('myElement');
      element.style.backgroundColor = 'blue';

      // Accessing the BOM to display an alert message
      alert('Background color changed!');
    }
  </script>
</head>
<body>
  <h1 id="myElement">Hello, BOM and DOM!</h1>
  <button onclick="changeBackgroundColor()">Change Background</button>
</body>
</html>
```

In the example above, the change Background Color function is triggered when clicking the button. It uses the DOM to access the element with the ID "my Element" and changes its background colour to blue. It also uses the BOM to display an alert message. This demonstrates the interaction between the BOM and DOM in a web page.

Identifying Selectors in JavaScript

Selectors in JavaScript are used to identify and select specific elements from the Document Object Model (DOM) based on certain criteria. They allow you to target and manipulate specific elements on a web page. Here are a few commonly used selectors in JavaScript:

1. **getElementById:**
   - Selects an element based on its unique ID attribute.

Example:

```javascript
var element = document.getElementById('myElement');
```

2. **getElementsByClassName:**
   - Selects elements based on their class name.
   - Returns a collection of elements with the specified class.

Example:

```javascript
var elements = document.getElementsByClassName('myClass');
```

3. **getElementsByTagName:**
   - Selects elements based on their tag name.
   - Returns a collection of elements with the specified tag.

Example:

```javascript
var elements = document.getElementsByTagName('div');
```

4. **querySelector:**

   - Selects the first element that matches a specified CSS selector.

   Example:

```javascript
var element = document.querySelector('.myClass');
```

5. **querySelectorAll:**

   - Selects all elements that match a specified CSS selector.
   - Returns a collection of elements that match the selector.

   Example:

```javascript
var elements = document.querySelectorAll('input[type="text"]');
```

6. **parentElement:**
   - Selects the parent element of a given element.

   Example:

```javascript
var parent = element.parentElement;
```

7. **nextElementSibling:**
   - Selects the next sibling element of a given element.

   Example:

```javascript
var sibling = element.nextElementSibling;
```

8. **previousElementSibling:**
   - Selects the previous sibling element of a given element.

Example:

```javascript
var sibling = element.previousElementSibling;
```

Applying BOM and DOM in JavaScript

BOM (Browser Object Model) and DOM (Document Object Model) are important concepts in JavaScript that allow you to interact with the browser and manipulate the structure and content of web pages. Here are some ways to apply BOM and DOM in JavaScript:

1. **Accessing Elements in the DOM:**
   - Use the **document.getElementById()** method to select an element by its ID.
   - Use the **document.getElementsByClassName()** method to select elements by their class name.
   - Use the **document.getElementsByTagName()** method to select elements by their tag name.
   - Use the **document.querySelector()** method to select the first element that matches a CSS selector.
   - Use the **document.querySelectorAll()** method to select all elements that match a CSS selector.
2. **Modifying Element Content:**
   - Use the **element.innerHTML** property to get or set the HTML content of an element.
   - Use the **element.textContent** property to get or set the text content of an element.
   - Use the **element.setAttribute()** method to set attributes of an element.
   - Use the **element.style** property to manipulate the CSS styles of an element.
3. **Handling Events:**
   - Use the **element.addEventListener()** method to attach event listeners to elements.
   - Use event handlers like **onclick**, **onmouseover**, etc., to handle specific events.

- Use the **event** object to access information about the event and its target element.

4. **Navigating the DOM:**
   - Use properties like **element.parentNode**, **element.nextSibling**, and **element.previousSibling** to navigate the DOM tree.
   - Use methods like **element.querySelector()** and **element.querySelectorAll()** to find elements within a specific context.
5. **Accessing BOM Properties:**
   - Use the **window** object to access properties like **window.location** to get information about the current URL.
   - Use the **window.alert()** method to display an alert box.
   - Use the **window.open()** method to open a new browser window.
6. **Manipulating Browser History:**
   - Use the **window.history** object to navigate through the browser history using methods like **history.back()** and **history.forward()**.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Applying BOM and DOM</title>
  <script>
    function changeBackground() {
      // Accessing the DOM to change the background color
      var element = document.getElementById('myElement');
      element.style.backgroundColor = 'blue';

      // Accessing the BOM to open a new window
      var newWindow = window.open('', '', 'width=400,height=200');
      newWindow.document.write('<h2>New Window</h2>');
    }
  </script>
</head>
<body>
  <h1 id="myElement">Applying BOM and DOM</h1>
  <button onclick="changeBackground()">Change Background and Open Window</bu
</body>
</html>
```

Integrate JavaScript

Writing JavaScript Code

JavaScript is a high-level programming language that is primarily used for adding interactivity and dynamic functionality to web pages. Writing JavaScript code involves understanding the syntax and structure of the language and using its built-in features and functions to perform various tasks. Here is a general guide on describing and writing JavaScript code:

1. **Syntax and Structure:**
    - JavaScript code is written inside script tags (<script>...</script>) in an HTML document, or in a separate .js file that is linked to the HTML document.
    - JavaScript code is executed sequentially, from top to bottom, unless there are control flow statements like loops or conditionals.
2. **Variables:**
    - Declare variables using the 'var', 'let', or 'const' keyword followed by the variable name.
    - Assign values to variables using the assignment operator (=).
    - Variables can store various data types such as numbers, strings, booleans, arrays, or objects.
3. **Functions:**
    - Declare functions using the 'function' keyword followed by the function name and parentheses. Example: function myFunction() { ... }
    - Functions can take parameters inside the parentheses and perform specific tasks or calculations.
    - Functions can return a value using the 'return' statement.
4. **Control Flow:**
    - Use if-else statements for conditional branching. Example: if (condition) { // Code to execute if the condition is true } else { // Code to execute if the condition is false }
    - Use loops like 'for' or 'while' to repeat code blocks multiple times.
5. **Events and Event Handlers:**
    - Use event handlers to execute JavaScript code in response to user actions or events on the web page.
    - Common events include button clicks, form submissions, mouse movements, or key presses.

- Attach event handlers to HTML elements using the 'addEventListener' method.

6. **DOM Manipulation:**
    - Access and manipulate HTML elements using JavaScript to dynamically change their content, style, or attributes.
    - Use methods like 'getElementById', 'querySelector', or 'querySelectorAll' to select elements.
    - Modify element properties or contents using the 'innerHTML', 'textContent', or 'setAttribute' methods.
7. **Error Handling:**
    - Use try-catch blocks to handle and manage errors in JavaScript code.
    - Catch and handle exceptions using the 'try' and 'catch' statements.
8. **External Libraries and APIs:**
    - Utilize external JavaScript libraries or APIs by including their script files in the HTML document.
    - Follow the documentation and guidelines provided by the library or API provider to integrate their functionality into your code.

HTML:

```html
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Example</title>
</head>
<body>
  <button id="greetBtn">Click Me!</button>
  <p id="greeting"></p>

  <script src="script.js"></script>
</body>
</html>
```

Writing JavaScript code involves creating scripts or programs that web browsers can execute to perform various tasks on web pages. Here's an example that demonstrates how to write JavaScript code:

Example: Let's say we want to create a simple JavaScript code that displays a greeting message when a button is clicked on a web page.

JavaScript (script.js):

```javascript
// Access the button element and greeting paragraph
const greetBtn = document.getElementById("greetBtn");
const greeting = document.getElementById("greeting");

// Add a click event listener to the button
greetBtn.addEventListener("click", function() {
  // Update the greeting text
  greeting.textContent = "Hello, World!";
});
```

In this example, we have an HTML file with a button and a paragraph element. The JavaScript code is written in a separate file called **script.js**, which is linked to the HTML file using the **<script>** tag.

Inside the JavaScript code, we first access the button and paragraph elements from the HTML using **document.getElementById()**. We assign them to variables **greetBtn** and **greeting**, respectively.

Next, we add a click event listener to the **greetBtn** button using the **addEventListener()** method. The event listener listens for a click event on the button and executes the provided callback function when the event occurs. Inside the callback function, we update the **textContent** property of the **greeting** element to display the "Hello, World!" message.

The JavaScript code is executed when the button is clicked, and the greeting message is displayed on the web page.

Self-Check Sheet 2 Develop a website using JavaScript

Short Questions:

1. What is the purpose of a variable in JavaScript?
2. How do you declare a function in JavaScript?
3. What is the purpose of a loop in JavaScript?
4. How do you write a conditional statement in JavaScript?
5. What is the role of switches in JavaScript?
6. How do you create an object in JavaScript?
7. How do you define an array in JavaScript?
8. How do you output a message or result in JavaScript?
9. What is the purpose of comments in JavaScript?
10. What are the different data types available in JavaScript?
11. What is the purpose of BOM (Browser Object Model) in JavaScript?
12. What is the purpose of DOM (Document Object Model) in JavaScript?
13. How can you identify selectors in JavaScript?
14. How can you apply BOM in JavaScript development?
15. How can you apply DOM in JavaScript development?

Multiple-Choice Questions:

1. Which JavaScript platform allows you to run JavaScript on the server-side?
   a. Node.js b. React.js c. Vue.js d. Angular.js
2. Which JavaScript library is commonly used for DOM manipulation and event handling?
   a. jQuery b. MooTools c. JS charts d. ImageFX
3. Which JavaScript library provides tools for creating interactive charts and graphs?
   a. jQuery b. MooTools c. JS charts d. ImageFX
4. Which JavaScript core component is responsible for data validation in web forms?
   a. Validation b. Variables c. Functions d. Objects
5. Which JavaScript core component is used to store and manipulate collections of data?
   a. Variables b. Functions c. Arrays d. Objects
6. Which of the following represents the structured representation of HTML/XML documents in JavaScript?

a. BOM b. DOM c. CSS d. AJAX

7. What does BOM stand for?
   a. Browser Object Model b. Basic Object Model
   c. Document Object Model d. Data Object Model
8. Which of the following is not a type of selector in JavaScript?
   a. ID selector b. Class selector c. Element selector d. Method selector

9. What is the purpose of applying BOM in JavaScript?
   a. To manipulate web page elements
   b. To handle browser-specific events
   c. To perform calculations and computations
   d. To format and style web pages

10. How can you apply DOM in JavaScript?
    a. Accessing and modifying browser properties
    b. Manipulating web page elements and their attributes
    c. Handling server requests and responses
    d. Creating animations and effects

Fill in the Gap Questions:

1. _____ is used to execute a block of code repeatedly until a certain condition is met.
2. _____ are used to perform different actions based on different conditions.
3. _____ is a JavaScript library that simplifies working with dates and times.
4. _____ is used to add interactivity and dynamic content to web pages.
5. _____ are used to store multiple values in a single variable.

6. BOM stands for _____ Object Model.

7. The _____ represents the structured representation of HTML/XML documents.

8. In JavaScript, selectors are used to _____ web page elements.

9. Applying BOM in JavaScript allows you to manipulate _____ properties and behaviors.

10. DOM is used in JavaScript to access and modify _____ elements and their attributes.

Answer Sheet 2 Develop a website using JavaScript

Short Questions:

1. Variables store and manipulate data in JavaScript.
2. Functions are blocks of code that perform a specific task.
3. Loops are used to repeat a block of code multiple times.
4. Conditions are used to perform different actions based on specific criteria.
5. Switches are used to select one of many code blocks to be executed.
6. Objects represent real-world entities and have properties and methods.
7. Arrays are used to store multiple values in a single variable.
8. Output displays information or results in the browser console or web page.
9. Comments are used to add notes or explanations to the code for better understanding.
10. JavaScript has several data types, including numbers, strings, booleans, objects, arrays, etc.
11. The purpose of BOM in JavaScript is to provide access to browser objects and properties.
12. The purpose of DOM in JavaScript is to represent the structured representation of HTML/XML documents and provide access to elements and their properties.
13. Selectors in JavaScript are used to identify and target specific elements on a web page.
14. BOM can be applied in JavaScript development to handle browser-specific events, manipulate the browser window, and access browser properties.
15. DOM can be applied in JavaScript development to manipulate web page elements dynamically, access and modify element properties, styles, and content.

Multiple-Choice Questions:

1. a. Node.js
2. a. jQuery
3. c. JS charts
4. a. Validation
5. c. Arrays
6. b. DOM
7. a. Browser Object Model
8. d. Method selector
9. a. To handle browser-specific events.
10. a. Manipulating web page elements and their attributes

Fill in the Gap Questions:

1. Loops
2. Conditions
3. Datejs
4. React.js
5. Arrays
6. Browser
7. DOM (Document Object Model)
8. Manipulate
9. Browser
10. HTML/XML

Job Sheet 2 Interactive Travel Planner Website

Job Title: Interactive Travel Planner Website

Job Description: Develop a dynamic and interactive travel planner website using JavaScript. The website will allow users to plan and customise their trips by selecting destinations, dates, and activities. Your role is implementing the JavaScript functionality to make the website user-friendly and engaging.

1. Project Setup:
    - Set up a development environment with a code editor (e.g., Visual Studio Code).
    - Create the project folder structure (HTML, CSS, JavaScript files).

2. Front-end Development: Create a visually appealing front-end for the travel planner website using HTML and CSS.

3. JavaScript Implementation: Implement JavaScript to add the following features:
    - Destination Selection: Users should be able to select multiple destinations for their trip.
    - Date Selection: Allow users to choose their trip's start and end dates.
    - Activity Selection: Provide a list of activities and allow users to add or remove them from their itinerary.
    - Budget Calculator: Calculate the estimated budget based on the selected destinations and activities.
    - User-Friendly Interface: Ensure the website is intuitive, responsive, and user-friendly.

Specification Sheet 2 Interactive Travel Planner Website

Job Title:

Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer/Laptop	Minimum Corei3 with 4GB RAM	No.	1
3	Software (Browser)	Latest Version	No.	1
4	Internet connections	High Speed	No.	1

Other Specifications:

1. Front-end Development:
    - HTML: Create structured HTML templates for the website's layout.
    - CSS: Style the website to enhance its visual appeal and user experience.
    - JavaScript: Implement JavaScript to add interactivity and functionality to the website.

2. JavaScript Features:
    - Destination Selection:
        - Allow users to select multiple destinations.
        - Display selected destinations in a visually pleasing manner.
    - Date Selection:
        - Enable users to choose start and end dates for their trip.
        - Validate date inputs to prevent invalid selections.
    - Activity Selection:
        - Provide a list of activities for users to add to their itinerary.
        - Allow users to add or remove activities from their plan.
    - Budget Calculator:
        - Calculate and display the estimated budget based on selected destinations and activities.
        - Update the budget in real-time as users make selections.
    - User-Friendly Interface:
        - Ensure a smooth and intuitive user experience.
        - Implement clear and user-friendly interface elements.

Learning Outcome 3: Test website

Assessment Criteria	1. The website is tested to ensure functionality and errors are corrected as per standard operating procedure. 2. The website is opened with common browsers and check for accessibility, readability, legibility and presentation in accordance with client requirements. 3. The website is evaluated for fitness in terms of the purpose, target audience and specifications of client requirements.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standards. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. Functional Testing and Error Correction 2. Cross-Browser Compatibility and Accessibility 3. Website Evaluation: Purpose, Audience, and Client Specifications
Training Methods	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience 3: Test website

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Students will ask the instructor about develop a website using JavaScript	1. The instructor will provide the learning materials for test website.
2. Read the **Information sheet/s**	2. Information Sheet No 3: Test website. ▪ Functional Testing and Error Correction ▪ Cross-Browser Compatibility and Accessibility ▪ Website Evaluation: Purpose, Audience, and Client Specifications
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No 3: Test website. Answer key No 3: Test website.
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Job Sheet No 3: Test website. Specification Sheet 3: Test website.

Information Sheet 3 Test website

Learning Objective: After completing this information sheet, the learners will be able to test website.

3.1 Testing and Error Correction
3.2 Cross-Browser Compatibility and Accessibility
3.3 Website Evaluation: Purpose, Audience, and Client Specifications

When it comes to developing a website using JavaScript, it's crucial to ensure that the final product looks great and functions seamlessly across various browsers and devices. Additionally, web accessibility should be a top priority to make your site inclusive and compliant with legal requirements. Let's explore the key aspects of website development with a focus on JavaScript:

3.1 Testing Website Functionality and Correcting Error

Functional testing is a critical phase in web development that ensures a website's interactive elements and features function as intended. This type of testing evaluates the website's behaviour, including its forms, buttons, navigation menus, and any dynamic content. The primary goal is to confirm that user interactions are smooth and that all features work consistently across various web browsers and devices.

- **Functional Testing:** Before anything else, it's vital to guarantee that your website functions as intended. This involves testing interactive elements like forms, buttons, navigation menus, and dynamic content. Ensure that user interactions are smooth and all features work across different browsers.
- **Error Identification:** It's natural for errors to occur during web development. These errors can range from broken links to misaligned elements and JavaScript bugs. Systematically identify and document these errors so that you can correct them efficiently.
- **Validation:** Use W3C Markup Service and CSS validation tools to check if your HTML and CSS code follows industry standards. Valid code is less prone to errors and functions better across various browsers.

<u>Key Objectives of Functional Testing:</u>

- **Verification of Functionality:** The core objective of functional testing is to verify that every interactive element on the website performs its intended function without errors or unexpected behavior. This includes features such as contact forms, login systems, search functionality, and interactive widgets.
- **Browser Compatibility:** Functional testing ensures that the website functions correctly on different web browsers like Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, and others. It's crucial because various browsers may interpret code differently, potentially leading to discrepancies in functionality.

- **Device Responsiveness:** Websites must be responsive and adaptive to different devices, including desktop computers, laptops, tablets, and smartphones. Functional testing assesses whether the website's interactive elements work seamlessly across various screen sizes and resolutions.

Key Steps in Functional Testing:

- **Test Planning:** Begin by creating a comprehensive test plan that outlines the scope of functional testing. Identify all interactive elements, features, and user flows that need to be tested. Specify the browsers and devices to be included in the testing process.
- **Test Case Design:** Develop detailed test cases for each interactive element and feature. Test cases should include step-by-step instructions for testing, expected outcomes, and criteria for success. Consider both positive and negative scenarios (expected user interactions) (error conditions).
- **Execution:** Execute the test cases on different browsers and devices as specified in the test plan. During testing, interact with the website as a user would and carefully observe the behavior of each element. Document any deviations from expected functionality.
- **Issue Reporting:** If any issues or defects are identified during testing, report them in a structured manner. Include detailed information about the issue, steps to reproduce it, the browser and device where it was encountered, and its impact on the user experience.
- **Regression Testing:** After issues are resolved, perform regression testing to ensure that fixes haven't introduced new problems. This step helps maintain the website's functionality as new features or changes are implemented.
- **Cross-browser Testing:** Test the website on various browsers, paying attention to differences in rendering and functionality. Use browser developer tools to debug and diagnose issues specific to particular browsers.
- **Device Testing:** Verify that the website functions correctly on different devices with varying screen sizes and resolutions. Emulators and real devices can be used to simulate mobile and tablet experiences.
- **Usability Testing:** As part of functional testing, assess the overall user experience. Ensure that the website's interactive elements are intuitive and user-friendly. Gather feedback on navigation, responsiveness, and overall satisfaction.
- **Accessibility Testing:** Functional testing should also consider accessibility. Ensure that interactive elements are accessible to users with disabilities, including those who rely on screen readers and keyboard navigation.

Checking Website Compatibility and Accessibility

- **Cross-browser Compatibility:** Test your website on multiple web browsers such as Chrome, Firefox, Safari, and Internet Explorer. Ensure that the website appears and functions consistently across these platforms. Use browser developer tools to identify and fix any compatibility issues.

- **Why it's Important:** Web users employ various browsers to access the internet, and each browser may interpret website code slightly differently. Ensuring cross-browser compatibility guarantees that your website looks and performs consistently, regardless of the browser used.

Testing Methodology:

- **Browser Selection:** Choose a set of commonly used web browsers, such as Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge, for testing.
- **Browser Developer Tools:** Utilize browser developer tools to inspect and debug your website on each browser. These tools provide insights into rendering issues, JavaScript errors, and layout problems.
- **Visual Inspection:** Manually inspect your website on different browsers to identify any visual discrepancies or layout issues. Pay attention to the alignment of elements, font rendering, and overall design consistency.

- **Responsive Design:** Verify that your website adapts gracefully to different screen sizes, from large desktop monitors to small mobile devices. This responsiveness is critical for providing a seamless user experience.

- **Why it's Important:** With the proliferation of various devices, your website must be responsive. Responsive design ensures that your site adapts gracefully to different screen sizes, providing an optimal user experience on desktop computers, laptops, tablets, and smartphones.

Testing Methodology:

- **Device Emulation:** Use device emulators or responsive design testing tools to simulate various screen sizes and resolutions. Verify that your website's layout adjusts appropriately.
- **Manual Testing:** Physically test your website on various devices and screen sizes. Interact with the site to confirm that all interactive elements, such as buttons and forms, remain usable and accessible.

- **Accessibility Testing:** Make sure your website is accessible to people with disabilities. Test with screen readers and keyboard navigation to ensure all users can navigate and interact with your content. Address any accessibility issues that you discover.

- **Why it's Important:** Web accessibility is about making your website usable for everyone, including individuals with disabilities. Ensuring accessibility is a legal requirement in many regions and a moral imperative.

Testing Methodology:

- **Screen Reader Testing:** Use screen reader software (e.g., JAWS, NVDA, VoiceOver) to navigate your website. Ensure that all content, including text, images, and interactive elements, is accessible through screen readers.
- **Keyboard Navigation:** Test your website's functionality using only a keyboard. Ensure that all interactive elements can be accessed and used without a mouse.
- **Alt Text for Images:** Verify that images have appropriate alternative text (alt text) that describes their content or purpose.
- **Semantic HTML:** Use semantic HTML elements (e.g., headings, lists, labels) to structure content logically. Ensure that form fields have associated labels.
- **Colour Contrast:** Check the colour contrast of text and background elements to ensure readability, especially for individuals with visual impairments.
- **Testing Tools:** Utilize accessibility testing tools and browser extensions (e.g., Axe, WAVE) to automate some aspects of accessibility testing and identify issues.

Evaluating Website Fitness for Purpose and Client Requirements

- **Client Requirements:** Review the initial client requirements and project objectives. Ensure that your website aligns with these requirements. Ensure all requested features are implemented, and the design matches the client's expectations.
- **Usability Testing:** Conduct usability testing with potential users or stakeholders. Gather feedback on the website's ease of use, clarity of content, and overall user experience. Make improvements based on this feedback.
- **Performance Evaluation:** Assess the website's loading speed and overall performance. Use tools like Google PageSpeed Insights to identify areas for improvement. A fast-loading website is essential for retaining visitors.
- **Content Review:** Ensure that all content, including text, images, and multimedia elements, is accurate, up-to-date, and relevant to the website's purpose. Broken or outdated content can negatively impact the user experience.

An example Testing Case/Scenario: Testing JavaScript Functionality, Compatibility, and Accessibility for an E-commerce Website

Background: ABC Electronics, a leading e-commerce company, is preparing to launch a new website to showcase and sell its latest tech products. JavaScript is pivotal in creating dynamic user interfaces and enhancing the shopping experience. The testing team ensures that the website's JavaScript features work flawlessly, are compatible across browsers and devices, and are accessible to all users.

Testing Objectives:

1. Verify that JavaScript-based features on the website function correctly.
2. Ensure cross-browser compatibility to provide a consistent experience.
3. Guarantee accessibility for all users, including those with disabilities.

Testing Steps:

Functional Testing (JavaScript Features):

1. **Product Filtering:** Test the functionality of product filtering using JavaScript. Verify that users can filter products by category, price range, and brand.
    - **Expected Outcome:** Users should be able to filter products and see updated results without errors.

2. **Shopping Cart:** Test the shopping cart functionality, including adding and removing items, updating quantities, and calculating totals.
    - **Expected Outcome:** Items should be added, updated, and removed accurately, and the cart total should reflect the changes correctly.

3. **User Authentication:** Verify the login and registration processes, ensuring users can create accounts, log in, and maintain their sessions.
    - **Expected Outcome:** Users should be able to register, log in, and perform actions associated with authenticated accounts without errors.

Cross-browser Compatibility Testing:

1. **Browser Selection:** Test the website on popular browsers, including Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge.
    - **Expected Outcome:** The website should render and function consistently across all tested browsers.

2. **Browser Developer Tools:** Use browser developer tools to inspect and resolve JavaScript-related issues.
    - **Expected Outcome:** Any JavaScript errors or issues specific to particular browsers should be identified and fixed.
3. **Visual Inspection:** Manually inspect the website on different browsers to identify visual discrepancies or layout issues.
    - **Expected Outcome:** The website's layout, including JavaScript-driven elements, should appear consistent across browsers.

Accessibility Testing:

1. **Screen Reader Testing:** Use screen reader software (e.g., NVDA) to navigate the website. Verify that all dynamic content and interactive elements are accessible.
    - **Expected Outcome:** Screen reader users should be able to interact with JavaScript features and receive appropriate feedback.

2. **Keyboard Navigation:** Test the website's functionality using keyboard navigation only. Ensure that all JavaScript-driven elements can be accessed and operated without a mouse.
    - **Expected Outcome:** Keyboard users should have a seamless experience interacting with JavaScript features.

3. **Alt Text for Images:** Check that images used in JavaScript features have appropriate alt text.
    - **Expected Outcome:** Images within dynamic content should have descriptive alt text.

4. **Semantic HTML:** Review the website's HTML structure to ensure proper semantic elements are used, especially for JavaScript-generated content.
    - **Expected Outcome:** JavaScript-driven content should be structured logically using semantic HTML.

5. **Colour Contrast:** Examine text and background colour contrast within JavaScript elements, ensuring readability for visually impaired users.
    - **Expected Outcome:** Text should have sufficient contrast to meet accessibility guidelines.

6. **Testing Tools:** Utilize accessibility testing tools (e.g., Axe, WAVE) to automate some aspects of accessibility testing and identify issues.
    - **Expected Outcome:** Automated tests should highlight any accessibility issues that require attention.

Upon successful testing and fixes, if required, any of JavaScript functionality, cross-browser compatibility, and accessibility, ABC Electronics can confidently launch their e-commerce website, providing a seamless and inclusive shopping experience for all users while showcasing their latest tech products.

Self-Check Sheet 3 Test website

1. What is the primary goal of functional testing in web development?
    - A. Checking website compatibility
    - B. Ensuring cross-browser compatibility
    - C. Verifying that interactive elements work as intended
    - D. Evaluating website fitness for purpose

2. During functional testing, what is the purpose of regression testing?
    - A. To identify errors systematically
    - B. To ensure cross-browser compatibility
    - C. To check responsiveness on mobile devices
    - D. To confirm that fixes haven't introduced new issues

3. Why is validation of JavaScript code essential in functional testing?
    - A. It ensures cross-browser compatibility.
    - B. It identifies visual discrepancies.
    - C. Valid code is less prone to errors.
    - D. It improves website performance.

4. What is the purpose of cross-browser compatibility testing?
    - A. To ensure that the website is accessible
    - B. To verify that the website functions as intended
    - C. To confirm that the website appears consistent on different browsers
    - D. To evaluate the website's loading speed

5. Why is responsive design testing important for a website?
    - A. To check for broken links
    - B. To ensure accessibility
    - C. To verify functionality
    - D. To adapt to different screen sizes

6. What is the main goal of accessibility testing?
    - A. To verify functionality
    - B. To ensure cross-browser compatibility
    - C. To make the website usable for people with disabilities
    - D. To assess website performance

7. In the context of website evaluation, what should you do to ensure the website aligns with client requirements?
    - A. Conduct usability testing
    - B. Review client feedback
    - C. Assess the loading speed
    - D. Review initial client requirements

8. What is the purpose of usability testing during website evaluation?
    - A. To assess loading speed
    - B. To gather feedback on the user experience
    - C. To identify broken links
    - D. To ensure cross-browser compatibility

9. Why is performance evaluation important for a website?
    - A. To ensure accessibility
    - B. To verify functionality
    - C. To assess loading speed and overall performance
    - D. To check for broken links

10. What should be reviewed to ensure that website content is accurate and relevant?
    - A. Browser compatibility
    - B. Code validation
    - C. Content alignment
    - D. Client feedback

Answer Sheet 3 Test website

1. C. Verifying that interactive elements work as intended
2. D. To confirm that fixes haven't introduced new issues
3. C. Valid code is less prone to errors.
4. C. To confirm that the website appears consistent on different browsers
5. D. To adapt to different screen sizes
6. C. To make the website usable for people with disabilities
7. D. Review initial client requirements
8. B. To gather feedback on the user experience
9. C. To assess loading speed and overall performance
10. C. Content alignment

Job Sheet 3 Test Website Cross-browser compatibility

Job Name: Test Website Cross-browser compatibility.

Procedure:

1. Choose a set of commonly used web browsers.
2. Manually inspect your website on different browsers to identify visual discrepancies or layout issues.
3. Utilise browser developer tools to inspect and debug your website on each browser.
4. Compatibility and Responsiveness Testing:
    - Test the website on different web browsers to ensure consistent functionality and appearance.
    - Use browser developer tools for debugging if needed.
    - Test the website on various devices to ensure responsiveness.
5. Accessibility Implementation:
    - Add keyboard navigation support for all interactive elements.
    - Use semantic HTML elements and proper labelling for forms.
    - Provide alt text for images and icons.
6. Testing and Quality Assurance:
    - Thoroughly test the website's functionality, ensuring all features work as expected.
    - Test different scenarios and edge cases to identify and fix potential issues.

Specification Sheet 3 Test Website Cross-browser compatibility

Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer	Minimum Corei3 with 4 GB RAM	No.	1
2	Code Editor	Any	No.	1
3	Web Browser for Testing	Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge	No.	1

Other Specifications

1. Use the Website developed in Job 2 for reference.

Review of Competency

Below is your assessment rating for module **Develop Website Using JavaScript**

Assessment of Performance Criteria	Yes	No
The purpose and intended audience of the website are identified.		
Functional requirements and constraints are identified.		
A coding plan is developed as required.		
The necessary software is installed, and functionality is checked.		
JavaScript is coded as per functional requirements.		
JavaScript files are integrated with website as per standard procedure.		
JavaScript is executed to finalize the website.		
The website is tested to ensure functionality and errors are corrected as per standard operating procedure.		
The website is opened with common browsers and check for accessibility, readability, legibility and presentation in accordance with client requirements.		
The website is evaluated for fitness in terms of the purpose, target audience and specifications of client requirements.		

I now feel ready to undertake my formal competency assessment.

Signed:

Date:

Development of CBLM:

The Competency Based Learning Material (CBLM) of '**Develop Website using JavaScript' (**Occupation: Web Design and Development for Freelancing, Level-3**)** for National Skills Certificate is developed by NSDA with the assistance of SIMEC System, ECF consultancy & SIMEC Institute JV (Joint Venture Firm) in the month of June 2023 under the contract number of package SD-9A dated 07th May 2023.

Sl No.	Name & Address	Designation	Contact number
1	Khondoker Ali Asgor Pavel	Writer	01711 873 008
2	Fatema	Editor	01912 464 747
3	Md. Amir Hossain	Co-Ordinator	01631 670 445
4	Mahbub Ul Huda	Reviewer	01735490491