Competency Based Learning Materials (CBLMs)

Web Design

Level-3

Module 5: Working With basic JavaScript & jQuery

Code: CBLM-ICT-WD-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 "Working with Basic JavaScript & jQuery" is developed based on NSDA approved Competency Standards and Competency Based Curriculum under Web Design Level-3 Occupation. It contains the information required to implement the Web Design 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 level-3 course.

Approved by

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

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

How to use this Competency Based Learning Materials (CBLMs)

The module, Working with Basic JavaScript & jQuery 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 practice the job. You may need to practice 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

Table of Contents

Module Content

Unit Title: Work with basic JavaScript & jQuery
Unit Code: OU- ICT-WD-05-L3-V1
Module Title: Working with basic JavaScript & jQuery

Module Descriptor: This module encompasses the necessary knowledge, skills, and attitudes (KAS) for establishing work with basic JavaScript & jQuery. It specifically includes identifying JavaScript Core, introducing BOM & DOM, integrating JavaScript, and integrating jQuery.

Nominal Hours: 40

Learning Outcomes:

Upon completion of this module the trainees will be able to:

1. Identify JavaScript Core
2. Introduce BOM & DOM
3. Integrate JavaScript
4. Integrate jQuery.

Assessment Criteria:

1. JavaScript core components are identified.
2. The basic JavaScript platform are identified.
3. JavaScript library is interpreted.
4. BOM (Browser Object Model) & DOM (Document Object Model) are interpreted.
5. Selectors are identified.
6. BOM & DOM are applied.
7. JavaScript is written.
8. JavaScript code is debugged.
9. JavaScript library is used.
10. jQuery is interpreted.
11. jQuery is integrated.
12. Commonly used jQuery functions are applied.

Learning Outcome 1: Identify JavaScript Core

Assessment Criteria	1. JavaScript core components are identified. 2. The basic JavaScript platform are identified. 3. JavaScript library is interpreted
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. JavaScript Core Components 2. The Basic JavaScript Platform 3. JavaScript Libraries
Learning Materials	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Training Methods	1. Discussion 2. Presentation 3. Demonstration 4. Guided Practice 5. Individual Practice 6. Project Work 7. Problem Solving 8. Brainstorming
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience: Identify JavaScript Core

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 work with basic JavaScript & jQuery	1. The instructor will provide the learning materials for` identify JavaScript Core.
2. Read the **Information sheet/s**	2. Information Sheet No: Identify JavaScript Core
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No: 1: Identify JavaScript Core Answer key No: 1: Identify JavaScript Core
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Job Sheet No: 1 Assist JavaScript Development Specification Sheet: 1 Assist JavaScript Development

Information Sheet 1: Identify JavaScript Core

Learning Objective:

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

1.1 JavaScript Core Components
1.2 Basic JavaScript Platform
1.3 JavaScript Libraries

1.1 JavaScript Core Components

1.1.1 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;
```

1.1.2 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;
}
```

1.1.3 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++;
}
```

1.1.4 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.");
}
```

1.1.5 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.");
}
```

1.1.6 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 + ".");
  }
};
```

1.1.7 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"];
```

1.1.8 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!");
```

1.1.9 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.
*/
```

1.1.10 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
```

1.1.11 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
```

1.1.12l. 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
```

1.1.13Breaks:

- 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);
}
```

1.1.14 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);
}
```

1.1.15 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.

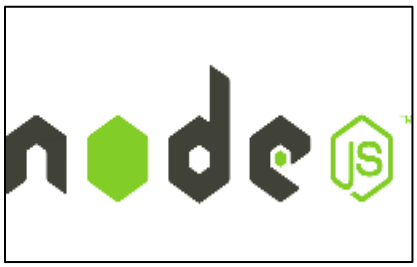
Example:

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

1.2 Basic JavaScript Platform

1.2.1 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.



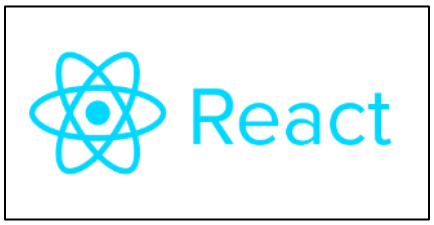
Example:

```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/');
});
```

1.2.2 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;
```

1.2.3 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>
```

1.2.4 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.

1.3 JavaScript Libraries

a. jQuery:

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

Example:

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

b. 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');
    }
  });
});
```

c. 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>
```

d. 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();
```

e. 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.

Self-Check Sheet – 1: Identify JavaScript Core

Short Questions:

1. What is the purpose of a variable in JavaScript?

Ans:

2. How do you declare a function in JavaScript?

Ans:

3. What is the purpose of a loop in JavaScript?

Ans:

4. How do you write a conditional statement in JavaScript?

Ans:

5. What is the role of switches in JavaScript?

Ans:

6. How do you create an object in JavaScript?

Ans:

7. How do you define an array in JavaScript?

Ans:

8. How do you output a message or result in JavaScript?

Ans:

9. What is the purpose of comments in JavaScript?

Ans:

10. What are the different data types available in JavaScript?

Ans:

Answer Key – 1: Identify JavaScript Core

Answers:

1. What is the purpose of a variable in JavaScript?

Ans: Variables store and manipulate data in JavaScript

2. How do you declare a function in JavaScript?

Ans:

3. What is the purpose of a loop in JavaScript?

Ans: Loops are used to repeat a block of code multiple times

4. How do you write a conditional statement in JavaScript?

Ans: Conditions are used to perform different actions based on specific criteria

5. What is the role of switches in JavaScript?

Ans: Switches are used to select one of many code blocks to be executed

6. How do you create an object in JavaScript?

Ans: Objects represent real-world entities and have properties and methods

7. How do you define an array in JavaScript?

Ans: Arrays are used to store multiple values in a single variable

8. How do you output a message or result in JavaScript?

Ans: Output is the display of information or results in the browser console or web page

9. What is the purpose of comments in JavaScript?

Ans: Comments are used to add notes or explanations to the code for better understanding

10. What are the different data types available in JavaScript?

Ans: JavaScript has several data types, including numbers, strings, booleans, objects, arrays, etc.

Job Sheet 1: Assist JavaScript Development

Objective: To gain hands-on experience and understanding of JavaScript core components, the basic JavaScript platforms, and interpreting JavaScript libraries.

Required Equipment:

1. Computer with internet access
2. Text editor or integrated development environment (IDE) for writing JavaScript code
3. Web browser for testing and debugging JavaScript code
4. Optional: JavaScript libraries (jQuery, MooTools, JS charts, ImageFX, Datejs) if you choose to explore them further

Procedure:

1. Familiarize yourself with the JavaScript core components, basic JavaScript platforms, and JavaScript libraries mentioned in the job sheet.
2. Set up your development environment by installing a text editor or IDE of your choice.
3. Ensure you have a modern web browser installed on your computer.
4. Start by exploring the JavaScript core components:
   - Review the list of components mentioned in the job sheet (variables, functions, loops, conditions, switches, objects, arrays, output, comments, data types, functions, operators, comparisons, breaks, errors, validation).
   - Research each component individually, understanding their purpose, syntax, and usage.
   - Practice writing code snippets that demonstrate the usage of each component.
5. Move on to understanding the basic JavaScript platforms:
   - Research and learn about Node.js, React.js, Vue.js, and Angular.js.
   - Understand their features, advantages, and use cases.
   - Explore sample code examples and tutorials related to each platform.
6. Proceed to interpreting JavaScript libraries:
   - Research and learn about jQuery, MooTools, JS charts, ImageFX, and Datejs.
   - Understand the purpose and functionality provided by each library.
   - Explore sample code examples and documentation to understand how to use each library effectively.
7. Practice integrating the JavaScript core components, platforms, and libraries into small projects or exercises to solidify your understanding.
8. Experiment with different code scenarios and test the outcomes using your web browser's developer tools.
9. Document your learning journey, noting down any challenges faced, interesting findings, and code examples that you find particularly useful.
10. Seek out online resources, tutorials, and forums to enhance your understanding and connect with the JavaScript development community.

Specification Sheet- 1 Assist JavaScript Development

Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer	Minimum Corei3 with 4GB RAM	Set	01
2	Web Browser (e.g., Google Chrome)	Latest Version	No.	01
3	Internet connections	High Speed	Set	01
4	Image processing software (e.g., Adobe Photoshop, GIMP)	Latest Version	No.	1

Learning Outcome 2: Introduce BOM & DOM

Assessment Criteria	1 BOM (Browser Object Model) & DOM (Document Object Model) are interpreted. 2 Selectors are identified 3 BOM & DOM are applied.
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 BOM (Browser Object Model) and DOM (Document Object Model) 2 Selectors in JavaScript 3 BOM and DOM in JavaScript
Learning Materials	1 CBLM 2 Handouts 3 Books, Manuals 4 Module/ Reference 5 Paper 6 Pen
Training Methods	1 Discussion 2 Presentation 3 Demonstration 4 Guided Practice 5 Individual Practice 6 Project Work 7 Problem Solving 8 Brainstorming
Assessment Methods	1 Written Test 2 Demonstration 3 Oral Questioning

Learning Experience 2: Introduce BOM & DOM

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 work with basic JavaScript & jQuery	1. The instructor will provide the learning materials for` introduce BOM & DOM.
2 Read the **Information sheet/s**	2. Information Sheet No: Introduce BOM & DOM
3 Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No: 2: Introduce BOM & DOM Answer key No. 2: Introduce BOM & DOM
4 Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet ▪ Job Sheet No: 2 Assist JavaScript Web Developer ▪ Specification Sheet: 2 Assist JavaScript Web Developer

Information Sheet 2: Introduce BOM & DOM

Learning Objective:

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

2.1 BOM (Browser Object Model) and DOM (Document Object Model)
2.2 Selectors in JavaScript
2.3 BOM and DOM in JavaScript

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

2.1.1 BOM (Browser Object Model):

- The Browser Object Model (BOM) represents the objects and interfaces provided by web browsers 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 standardized, and different browsers may implement different features or have variations in their implementation.

2.1.2 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 DOM is standardized by the World Wide Web Consortium (W3C), ensuring a consistent interface across different browsers.

2.1.3 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, when the button is clicked, the change Background Color function is triggered. It uses the DOM to access the element with the ID "my Element" and changes its background color 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.

2.2 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:

2.2.1 getElementById:

- Selects an element based on its unique ID attribute.

Example:

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

2.2.2 getElementsByClassName:

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

Example:

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

2.2.3 getElementsByTagName:

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

Example:

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

2.2.4 querySelector:

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

Example:

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

2.2.5 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"]');
```

2.2.6 parentElement:

- Selects the parent element of a given element.

Example:

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

2.2.7 nextElementSibling:

- Selects the next sibling element of a given element.

Example:

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

2.2.8 previousElementSibling:

- Selects the previous sibling element of a given element.

Example:

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

These selectors allow you to retrieve specific elements from the DOM and perform operations on them using JavaScript. They are often used in conjunction with other JavaScript methods and functions to manipulate and interact with the selected elements.
Note: Selectors can be used in both vanilla JavaScript and JavaScript libraries/frameworks like jQuery, allowing for more convenient and powerful element selection and manipulation.

2.3 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()**.

These are just a few examples of how you can apply BOM and DOM in JavaScript. By leveraging these concepts, you can create interactive web pages, handle user actions, modify content dynamically, and interact with various browser features.

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="myE
  <button onc
</body>
</html>
```

Regenerate response

Self-Check Sheet 2: Introduce BOM & DOM

Short Questions:

1. What is the purpose of BOM (Browser Object Model) in JavaScript?

   Answer:

2. What is the purpose of DOM (Document Object Model) in JavaScript?

   Answer:

3. How can you identify selectors in JavaScript?

   Answer:

4. How can you apply BOM in JavaScript development?

   Answer:

5. How can you apply DOM in JavaScript development?

   Answer:

Multiple-Choice Questions:

1. Which of the following represents the structured representation of HTML/XML documents in JavaScript?
   a. BOM b. DOM c. CSS d. AJAX

2. What does BOM stand for?
   a. Browser Object Model b. Basic Object Model
   c. Document Object Model d. Data Object Model

3. Which of the following is not a type of selector in JavaScript?
   a. ID selector b. Class selector c. Element selector d. Method selector

4. 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

5. 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

Answer Key 2: Introduce BOM & DOM

Answers: Short Questions:

1. The purpose of BOM in JavaScript is to provide access to browser objects and properties.
2. The purpose of DOM in JavaScript is to represent the structured representation of HTML/XML documents and provide access to elements and their properties.
3. Selectors in JavaScript are used to identify and target specific elements on a web page.
4. BOM can be applied in JavaScript development to handle browser-specific events, manipulate the browser window, and access browser properties.
5. 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. b. DOM
2. Browser Object Model
3. d. Method selector
4. To handle browser-specific events
5. Manipulating web page elements and their attributes

Job Sheet 2: Assist JavaScript Web Developer

Objective: The objective of this job is to gain a comprehensive understanding of BOM (Browser Object Model) and DOM (Document Object Model), identify selectors in JavaScript, and apply BOM and DOM concepts in JavaScript development. This job will provide hands-on experience in working with the browser environment and manipulating web page elements using JavaScript.

Required Equipment:

1. Computer with internet access
2. Text editor or integrated development environment (IDE) for writing JavaScript code
3. Web browser for testing and debugging JavaScript code
4. Optional: Development tools such as browser developer consoles or debugging extensions

Procedure:

1. **Familiarize yourself with the concepts of BOM and DOM:**
    - Research and understand the purpose and structure of the BOM and DOM.
    - Learn how BOM represents browser objects and properties, such as window, document, location, history, etc.
    - Understand how DOM represents the structured representation of HTML/XML documents and provides access to elements and their properties.
2. **Explore selectors in JavaScript:**
    - Identify different types of selectors in JavaScript, such as element selectors, class selectors, ID selectors, attribute selectors, etc.
    - Research and understand how to select and manipulate elements using selectors in JavaScript.
    - Practice writing code snippets that demonstrate the usage of selectors to interact with web page elements.
3. **Apply BOM and DOM in JavaScript:**
    - Learn how to access and modify browser properties and behaviors using BOM.
    - Understand how to access and manipulate HTML elements and their attributes using DOM.
    - Practice writing code that interacts with the browser environment and manipulates web page elements dynamically.
4. **Experiment with different BOM and DOM functionalities:**
    - Explore methods for manipulating the browser window, such as opening new windows, resizing, and navigating.
    - Learn how to access and modify element properties, styles, and content using DOM.
    - Practice handling events and implementing interactivity on web pages using JavaScript.
5. **Test and debug your JavaScript code:**

- Use the web browser's developer tools and debugging features to test and debug your JavaScript code.
- Practice inspecting and manipulating elements in real-time using the browser's console.

6. **Document your learning journey:**
    - Take notes and document your understanding of BOM, DOM, selectors, and their application in JavaScript.
    - Create a code repository to store and organize your code examples and projects.
7. **Seek additional resources and practice:**
    - Explore online tutorials, documentation, and articles on BOM, DOM, and JavaScript selectors.
    - Engage in coding exercises and projects to reinforce your understanding and skills.
8. **Share your findings and insights:**
    - Present your learnings to peers or mentors, explaining the concepts and showcasing your code examples.
    - Engage in discussions and collaborate with others in online forums or coding communities.

Note: The procedure above provides a general guideline for learning and applying the mentioned topics. Adjust and customize the process according to your learning style and available resources.

Specification Sheet-2 Assist JavaScript Web Developer

Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer	Minimum Corei3 with 4GB RAM	Set	01
2	Web Browser (e.g., Google Chrome)	Latest Version	No.	01
3	Internet connections	High Speed	Set	01
4	Image processing software (e.g., Adobe Photoshop, GIMP)	Latest Version	No.	1

Learning Outcome 3: Integrate JavaScript

Assessment Criteria	1. JavaScript is written. 2. JavaScript code is debugged. 3. JavaScript library is used.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standard. 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. Writing procedure of JavaScript Code. 2. Debugging process JavaScript Code. 3. JavaScript Libraries.
Learning Materials	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Training Methods	1. Discussion 2. Presentation 3. Demonstration 4. Guided Practice 5. Individual Practice 6. Project Work 7. Problem Solving 8. Brainstorming
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience 3: Integrate 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 work with JavaScript & jQuery	1. The instructor will provide the learning materials for integrate JavaScript.
2 Read the **Information sheet/s**	2. Information Sheet No: 3 Integrate JavaScript
3 Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No:3: Integrate JavaScript Answer key No.3: Integrate JavaScript
4 Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Job Sheet No:3 Write JavaScript Code, Debugging JavaScript Code and Utilizing JavaScript Libraries Specification Sheet: 3 Write JavaScript Code, Debugging JavaScript Code and Utilizing JavaScript Libraries

Information Sheet 3: Integrate JavaScript

Learning Objective:

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

3.1 Writing JavaScript Code.
3.2 Debugging JavaScript Code.
3.3 Utilizing JavaScript Libraries.

3.1 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:

3.1.1 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.

3.1.2 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.

3.1.3 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.

3.1.4 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.

3.1.5 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.

3.1.6 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.

3.1.7 Error Handling:

- Use try-catch blocks to handle and manage errors in JavaScript code.
- Catch and handle exceptions using the 'try' and 'catch' statements.

3.1.8 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 can be executed by web browsers 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.

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

Note: JavaScript code can be embedded directly in HTML using **<script>** tags or placed in separate external JavaScript files and linked to HTML using the **src** attribute. The example above demonstrates the latter approach.

3.2 Debugging JavaScript Code

Debugging JavaScript code is an essential skill for identifying and fixing errors or issues in your code. Here are some common techniques and tools for debugging JavaScript code:

3.2.1 Using console.log():

- Place **console.log()** statements in your code to output specific values or messages to the browser's console. This helps you track the flow of your code and verify the values of variables at different points.

3.2.2 Utilizing breakpoints:

- Modern web browsers come with built-in developer tools that allow you to set breakpoints in your JavaScript code. A breakpoint is a designated spot where the code execution pauses, allowing you to inspect variables, step through the code line by line, and identify the cause of any issues.

3.2.3 Inspecting variables and values:

- When your code is paused at a breakpoint, you can use the developer tools to inspect the values of variables, objects, and arrays. This helps you verify if the values are correct and identify any unexpected behavior.

3.2.4 Step-by-step execution:

- While debugging, you can use the step-by-step execution feature of the developer tools to navigate through your code line by line. This helps you understand the flow of execution and pinpoint the exact line where an error occurs.

3.2.5 Error messages and stack trace:

- When an error occurs in your JavaScript code, the browser's console displays an error message along with a stack trace. The error message provides information about the type of error and the line number where it occurred. The stack trace shows the sequence of function calls leading to the error, helping you trace the problem back to its source.

3.2.6 Debugging tools:

- Web browsers have powerful debugging tools that provide a range of features for debugging JavaScript code. These tools include the Console, Debugger, Sources, and Network panels, which allow you to inspect variables, set breakpoints, monitor network requests, and more.

Remember to remove or disable any debugging statements (e.g., console.log()) once you have resolved the issues in your code to ensure optimal performance.
By effectively using these debugging techniques and tools, you can identify and fix errors in your JavaScript code, resulting in more reliable and functional web applications.

3.3 Utilizing JavaScript Libraries

Utilizing JavaScript libraries can greatly enhance your web development process by providing pre-built functions, components, and utilities that simplify complex tasks and streamline your code. Here are some popular JavaScript libraries and their applications:

3.3.1 jQuery:

- jQuery is a fast and lightweight library that simplifies HTML document traversal, event handling, animation, and AJAX interactions. It allows you to write concise and cross-browser-compatible code. Example: **$('.my-element').addClass('active');**

3.3.2 React.js:

- React.js is a powerful library for building user interfaces. It allows you to create reusable UI components and efficiently update the UI when data changes. React uses a virtual DOM for performance optimization.

Example:

```javascript
function App() {
  return <h1>Hello, React!</h1>;
}
```

3.3.3 Vue.js:

- Vue.js is a progressive JavaScript framework for building user interfaces. It emphasizes simplicity and ease of integration. Vue offers features such as component-based architecture, two-way data binding, and declarative rendering.

Example:

```javascript
new Vue({
  el: '#app',
  data: {
    message: 'Hello, Vue!'
  }
});
```

3.3.4 Angular.js:

- Angular.js is a comprehensive framework for building dynamic web applications. It provides a powerful MVC (Model-View-Controller) architecture, two-way data binding, dependency injection, and extensive tooling.

Example:

```javascript
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
  $scope.message = 'Hello, Angular!';
});
```

3.3.5 D3.js:

- D3.js is a data visualization library that enables you to create interactive and dynamic charts, graphs, and maps. It provides powerful data-driven transformations and DOM manipulation capabilities. Example:

Example:

```javascript
var data = [10, 20, 30, 40, 50];
d3.select("body").selectAll("p")
  .data(data)
  .enter()
  .append("p")
  .text(function(d) { return "Value: " + d; });
```

When utilizing JavaScript libraries, it is important to consider factors such as the library's functionality, performance, community support, compatibility with your project, and any additional dependencies.

Always refer to the library's documentation and follow best practices to effectively integrate and utilize the library within your project. Additionally, ensure that you stay up-to-date with library updates and security patches to maintain the stability and security of your application.

Self-Check Sheet 3: Integrate JavaScript

Short Questions:

1. What is the purpose of writing JavaScript code?

   Answer:

2. Why is debugging an important aspect of JavaScript development?

   Answer:

3. How can JavaScript libraries enhance the development process?

   Answer:

4. What are some best practices to follow when writing JavaScript code?

   Answer:

5. What are some common debugging techniques in JavaScript?

   Answer:

Fill in the Gap Questions:

1. JavaScript code is typically written within _____ tags in an HTML document.
2. The process of identifying and fixing errors in the code is called _____.
3. JavaScript libraries are pre-written collections of _____ that provide additional functionality.
4. A _____ is a block of reusable code that performs a specific task.
5. The _____ method is used to add new elements to an array.

Answer Key 3: Integrate JavaScript

1. What is the purpose of writing JavaScript code?

   Answer: The purpose of writing JavaScript code is to create interactive and dynamic functionality in web applications

2. Why is debugging an important aspect of JavaScript development?

   Answer: Debugging is important to identify and fix errors in the code, ensuring the code runs as intended

3. How can JavaScript libraries enhance the development process?

   Answer: JavaScript libraries enhance development by providing pre-built functions, components, and utilities that can be easily integrated into projects, saving time and effort

4. What are some best practices to follow when writing JavaScript code?

   Answer: Some best practices when writing JavaScript code include using descriptive variable names, adding comments for clarity, organizing code into functions, and following coding conventions

5. What are some common debugging techniques in JavaScript?

   Answer: Common debugging techniques in JavaScript include using console.log() statements, setting breakpoints, examining error messages, and stepping through the code

Fill in the Gap Questions:

1. <script> tags
2. debugging
3. functions
4. function
5. push

Job Sheet 3: Write JavaScript Code, Debugging JavaScript Code and Utilizing JavaScript Libraries

Job Name: JavaScript Developer Objective: To write, debug, and utilize JavaScript code to develop interactive and functional web applications.

Required Equipment:

1. Computer or laptop with internet access
2. Web browser (Chrome, Firefox, etc.)
3. Integrated Development Environment (IDE) or code editor (e.g., Visual Studio Code, Sublime Text)
4. Browser developer tools (built-in tools in Chrome, Firefox, etc.)
5. Relevant JavaScript libraries and frameworks (based on project requirements)

Procedure:

1. **Writing JavaScript Code:** a. Analyze project requirements and functional specifications. b. Plan and design the JavaScript code structure and logic. c. Write clean, efficient, and maintainable JavaScript code following coding best practices. d. Test the code for functionality, handling edge cases, and error handling. e. Optimize and refactor the code for performance improvement. f. Document the code using meaningful comments for better understanding and maintainability. g. Collaborate with other team members (designers, backend developers) as needed.

2. **Debugging JavaScript Code:** a. Use browser developer tools to inspect and debug JavaScript code. b. Set breakpoints at specific lines to pause code execution for inspection. c. Examine variable values, objects, and arrays to identify issues. d. Use console.log() statements to output specific values and messages for debugging purposes. e. Step through the code line by line to understand the flow of execution. f. Analyze error messages and stack traces to pinpoint the source of errors. g. Fix identified issues and test the code to ensure proper functionality.

3. **Utilizing JavaScript Libraries:** a. Identify the requirements of the project and the specific functionalities that can be achieved using JavaScript libraries. b. Research and select suitable JavaScript libraries based on project needs and compatibility. c. Integrate the selected libraries into the project by including their script files or using package managers (e.g., npm, yarn). d. Understand the documentation and APIs of the libraries for proper usage. e. Utilize the provided functions, components, or utilities of the libraries to simplify and enhance the development process. f. Test the integrated libraries for compatibility and functionality. g. Stay updated with library updates and security patches to maintain the stability and security of the application.

Specification Sheet- 3 Write JavaScript Code, Debugging JavaScript Code and Utilizing JavaScript Libraries

Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer	Minimum Corei3 with 4GB RAM	Set	01
2	Web Browser (e.g., Google Chrome)	Latest Version	No.	01
3	Internet connections	High Speed	Set	01
4	Image processing software (e.g., Adobe Photoshop, GIMP)	Latest Version	No.	1

Learning Outcome 4: Integrate jQuery

Assessment Criteria	1. jQuery is interpreted 2. jQuery is integrated 3. Commonly used jQuery functions are applied
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standard. 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. jQuery 2. jQuery into Webpages 3. Commonly Used jQuery Functions
Learning Materials	1 CBLM 2 Handouts 3 Books, Manuals 4 Module/ Reference 5 Paper 6 Pen
Training Methods	1 Discussion 2 Presentation 3 Demonstration 4 Guided Practice 5 Individual Practice 6 Project Work 7 Problem Solving 8 Brainstorming
Assessment Methods	1 Written Test 2 Demonstration 3 Oral Questioning

Learning Experience 4: Integrate 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 work with basic JavaScript & jQuery	1. The instructor will provide the learning materials for integrate JavaScript
2 Read the **Information sheet/s**	2. Information Sheet No: 4 Integrate JavaScript
3 Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No: 4 Integrate JavaScript Answer key No 4: Integrate JavaScript
4 Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Job Sheet No: 4 – Integrate jQuery Specification Sheet: 4

Information Sheet 4: Integrate JavaScript

Learning Objective:

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

4.1 Interpreting jQuery
4.2 Integrating jQuery into Webpages
4.3 Applying Commonly Used jQuery Functions

4.1 Interpreting jQuery

Interpreting jQuery refers to understanding and using the jQuery library, which is a fast and concise JavaScript library designed to simplify HTML document traversal, event handling, and animation. It provides a convenient way to interact with the Document Object Model (DOM) and perform various operations on web elements.
Requirements:

- Computer or laptop with internet access
- Web browser (Chrome, Firefox, etc.)
- Integrated Development Environment (IDE) or code editor (e.g., Visual Studio Code, Sublime Text)
- jQuery library (can be downloaded from the jQuery website or referenced from a CDN)

Description: Interpreting jQuery involves understanding the syntax, functions, and capabilities of the jQuery library. jQuery allows developers to write less code while achieving more in terms of manipulating the DOM, handling events, making AJAX requests, and creating dynamic and interactive web pages.

Example: Here's a simple example that demonstrates the usage of jQuery to manipulate the DOM and handle events:

```html
<!DOCTYPE html>
<html>
<head>
  <title>jQuery Example</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script>
    // jQuery code
    $(document).ready(function() {
      // DOM manipulation
      $("#myButton").click(function() {
        $("#myText").text("Button clicked!");
      });
    });
  </script>
</head>
<body>
  <h1>jQuery Example</h1>
  <p id="myText">Hello, World!</p>
  <button id="myButton">Click Me</button>
</body>
</html>
```

4.1.1 Key Features:

DOM Manipulation: jQuery provides a simplified syntax to manipulate the DOM, allowing you to select, modify, and traverse HTML elements easily.

- Event Handling: jQuery simplifies event handling by providing methods to attach event listeners and perform actions when events occur (e.g., click, hover).
- AJAX Support: jQuery has built-in AJAX capabilities, making it easy to send and receive data from the server without refreshing the entire page.
- Animation Effects: jQuery includes methods to create smooth animations and transitions, such as fading elements in and out or sliding them across the screen.
- Plugin Architecture: jQuery has a vast ecosystem of plugins created by the community, extending its functionality for various purposes like sliders, carousels, form validation, and more.

4.1.2 Syntax:

- jQuery uses the $ symbol as a shorthand for the jQuery object. For example, $(selector) is used to select HTML elements.
- jQuery methods often take a callback function as a parameter, which is executed when the method is called or an event occurs.
- The chaining syntax allows you to perform multiple operations on the selected elements in a single statement.

4.1.3 Selectors:

- jQuery selectors are based on CSS selectors, allowing you to target elements by their tag name, class, ID, attributes, or hierarchical relationships.
- Common selectors include element selectors (e.g., $("p")), class selectors (e.g., $(".my-class")), ID selectors (e.g., $("#my-id")), and attribute selectors (e.g., $("[data-attribute='value']")).

4.1.4 Manipulating the DOM:

- jQuery provides methods to add, remove, modify, and traverse HTML elements.
- Methods like append(), prepend(), after(), and before() allow you to add new elements to the DOM.
- Methods like text(), html(), addClass(), removeClass(), and attr() help modify the content and attributes of elements.

4.1.5 Event Handling:

- jQuery offers a variety of methods to handle events, such as click(), hover(), submit(), and keydown().
- Event delegation can be used to handle events for dynamically created elements or elements that may not exist when the page loads.

- Methods like on(), off(), and trigger() provide more flexibility in event handling.

4.1.6 AJAX:

- jQuery simplifies AJAX requests with methods like $.ajax(), $.get(), $.post(), and $.getJSON().
- You can specify callback functions to handle success, error, and complete events when making AJAX requests.
- jQuery provides shorthand methods like $.load() to load HTML content dynamically into elements.
- Remember, practicing and experimenting with jQuery in real-world projects will deepen your understanding and proficiency with the library.

4.2 Integrating jQuery into Webpages

To integrate jQuery into webpages, follow these steps:

Step 1: Download or include the jQuery library

- You can download the jQuery library from the official website (https://jquery.com/download/) and save it in your project directory.
- Alternatively, you can use a content delivery network (CDN) to include jQuery directly in your HTML code.

For example:

```html
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

Step 2: Link the jQuery library in your HTML file

- Add the script tag to link the jQuery library in the head or body section of your HTML file.

```html
<script src="path/to/jquery.min.js"></script>
```

Step 3: Start using jQuery

- Once the jQuery library is linked, you can start using jQuery code in your HTML file. For example:

For example:

```html
<script>
  // jQuery code goes here
  $(document).ready(function() {
    // Your jQuery code here
  });
</script>
```

Step 4: Write jQuery code

- Inside the **$(document).ready()** function, write your jQuery code. This ensures that the code is executed only after the DOM is fully loaded.
- You can use jQuery selectors to target HTML elements and apply various methods to manipulate the DOM, handle events, perform animations, make AJAX requests, and more.

In this example, clicking the "Click Me" button changes its text to "Button Clicked!" using jQuery.

Remember to customize and expand the jQuery code to meet your specific requirements. You can refer to the jQuery documentation (https://api.jquery.com/) for a comprehensive list of methods and functionalities available.

4.3 Applying Commonly Used jQuery Functions

jQuery is a fast, lightweight, and feature-rich JavaScript library that simplifies the process of manipulating HTML documents, handling events, creating animations, and making AJAX requests. It provides a simple and concise syntax that allows developers to write code more efficiently and effectively.

Here are some key points to understand about jQuery:

- **DOM Manipulation:** jQuery makes it easy to select and manipulate elements in the Document Object Model (DOM) using CSS-style selectors. You can modify the content, attributes, styles, and position of elements on the page.

- **Event Handling:** jQuery simplifies event handling by providing methods like **.click()**, **.keydown()**, **.hover()**, etc. You can easily attach event handlers to elements and respond to user interactions such as clicks, mouse movements, keyboard input, etc.

- **Animation Effects:** jQuery provides built-in animation functions like **.fadeIn()**, **.fadeOut()**, **.slideDown()**, **.slideUp()**, etc., which allow you to create smooth and visually appealing effects on your webpages. You can animate properties like opacity, width, height, and position.
- **AJAX Support:** jQuery's AJAX functions, such as **$.ajax()**, **$.get()**, and **$.post()**, enable you to retrieve data from a server without reloading the entire page. You can send and receive data in various formats like JSON, XML, HTML, etc.

- **Cross-Browser Compatibility:** jQuery takes care of browser inconsistencies and provides a consistent API across different browsers. It abstracts away many browser-specific implementation details, allowing you to write code that works consistently across multiple browsers.

- **Extensibility:** jQuery can be extended through plugins to add additional functionality and features. There is a vast ecosystem of jQuery plugins available that can be easily integrated into your projects.

- **Community and Documentation:** jQuery has a large and active community of developers who contribute plugins, provide support, and share knowledge. The official jQuery website (https://jquery.com/) offers comprehensive documentation, tutorials, examples, and a forum for discussions.

jQuery provides a wide range of commonly used functions that make it easier to manipulate the DOM, handle events, perform animations, and make AJAX requests. Here are descriptions of some commonly used jQuery functions:

- **$(document).ready()**: This function ensures that your jQuery code executes only after the DOM has fully loaded. It takes a callback function as an argument, which will be executed when the DOM is ready.

Example:

```javascript
$(document).ready(function() {
  // Your code here
});
```

- **$(selector)**: This function selects one or more elements in the DOM based on the provided selector. It returns a jQuery object that represents the selected element(s).

Example:

```javascript
$("p") // Selects all <p> elements
$("#myElement") // Selects the element with ID "myElement"
$(".myClass") // Selects all elements with class "myClass"
```

- **.css():** This function is used to get or set CSS properties of selected elements. It accepts a property name as the first argument and can take additional arguments for property values.

Example:

```javascript
$("h1").css("color", "red"); // Sets the color of all <h1> elements to r
var fontSize = $("h1").css("font-size"); // Gets the font size of the fi
```

- **.addClass() and .removeClass():** These functions are used to add or remove CSS classes from selected elements.

Example:

```javascript
$("div").addClass("highlight"); // Adds the class "highlight" to all <div
$("p").removeClass("active"); // Removes the class "active" from all <p>
```

- **.text() and .html():** These functions are used to get or set the text or HTML content of selected elements.

Example:

```javascript
$("p").text("Hello!"); // Sets the text content of all <p> elements to
var htmlContent = $("div").html(); // Gets the HTML content of the firs
```

- **.click()**: This function attaches an event handler to selected elements for the "click" event. The provided callback function will be executed when the element is clicked.

```javascript
$("button").click(function() {
  alert("Button clicked!");
});
```

Example:

- .**fadeIn**() and .fadeOut(): These functions are used to create fading animations on selected elements.

Example:

```javascript
$("div").fadeIn(); // Fades in all <div> elements
$("img").fadeOut(); // Fades out all <img> elements
```

- $.ajax(): This function is used to make AJAX requests to a server and retrieve data asynchronously. It allows you to perform operations like retrieving JSON data, submitting forms, and updating parts of a webpage without reloading the entire page.

Example:

```javascript
$.ajax({
  url: "data.json",
  method: "GET",
  success: function(data) {
    console.log(data);
  },
  error: function(error) {
    console.log("Error: " + error);
  }
});
```

Self-Check Sheet 4: Integrate JavaScript

Short Questions:

1. What is jQuery?

   Answer:

2. How can you include jQuery in your web page?

   Answer:

3. What is the purpose of jQuery's .click() function?

   Answer:

4. What is the purpose of jQuery's .fadeOut() function?

   Answer:

5. What is the significance of the $(document).ready() function in jQuery?

   Answer:

Fill in the Gap Questions:

1. The $(selector) function in jQuery is used to select _____ elements in the DOM.
2. The .text() function in jQuery is used to get or set the _____ content of selected elements.
3. The .animate() function in jQuery is used to create _____ on selected elements.
4. The $.get() function in jQuery is used to perform a _____ request to retrieve data from a server.
5. The $(document).ready() function is used to ensure that the jQuery code executes only when the _____ has fully loaded.

Answer Key 4: Integrate JavaScript

Short Questions:

1. jQuery is a JavaScript library that simplifies HTML document traversal, event handling, animation, and AJAX interactions.
2. You can include jQuery in your web page by adding the jQuery script tag before your own JavaScript code.
3. The .click() function is used to attach a click event handler to selected elements in jQuery.
4. The .fadeOut() function is used to create a fading animation by gradually reducing the opacity of selected elements.
5. The $(document).ready() function ensures that the jQuery code executes only after the DOM has fully loaded.

Fill in the Gap Questions:

1. Answer: HTML
2. Answer: text
3. Answer: animations
4. Answer: GET
5. Answer: DOM

Job Sheet 4: Integrate jQuery

Objective: The objective of this job is to interpret jQuery, integrate jQuery into webpages, and apply commonly used jQuery functions to enhance the functionality and interactivity of webpages.

Required Equipment:

1. Computer with internet access
2. Text editor or integrated development environment (IDE)
3. Web browser

Procedure:

1. Study and understand the concepts and principles of jQuery, including its syntax, features, and benefits.
2. Familiarize yourself with the various ways to integrate jQuery into webpages, such as including the jQuery library via a script tag or using a package manager.
3. Learn and practice using commonly used jQuery functions, such as selecting elements, manipulating DOM elements, handling events, making AJAX requests, and creating animations.
4. Gain knowledge of jQuery plugins and their usage to extend the functionality of jQuery.
5. Experiment with jQuery code snippets and examples to reinforce your understanding of the concepts.
6. Create a sample webpage and apply jQuery to enhance its interactivity and functionality.
7. Test and debug your jQuery code to ensure it functions as intended.
8. Document your work, including code comments and explanations for future reference.
9. Stay updated with the latest version of jQuery and its features by referring to official documentation, online resources, and community forums.
10. Continuously improve your jQuery skills by exploring advanced topics, best practices, and emerging trends in jQuery development.

Note: The above procedure serves as a general guideline. Specific tasks and requirements may vary depending on the project or job context.

Specification Sheet- 4: Integrate jQuery

Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer	Minimum Corei3 with 4GB RAM	Set	01
2	Web Browser (e.g., Google Chrome)	Latest Version	No.	01
3	Internet connections	High Speed	Set	01
4	Image processing software (e.g., Adobe Photoshop, GIMP)	Latest Version	No.	1

Review of Competency

Below is your assessment rating for module **work with basic JavaScript & jQuery**

Assessment of Performance Criteria	Yes	No
Image processing software is identified.		
Basic shape is created.		
Image manipulation is performed.		
Web UI (User Interface) is created.		
Graphic design object is identified.		
Image processing software is selected.		
Smart objects vs normal layers are introduced.		
Layers are used.		
Objects are sliced.		
Objects are exported for web.		
Graphic assets are integrated to webpage.		

I now feel ready to undertake my formal competency assessment.

Signed:

Date:

Reference

https://www.google.com/search?q=Step+by+steoExport+the+Object+Adobe+Photoshop+-+wikihow&tbm=isch&ved=2ahUKEwiN98mI39GBAxWgz6ACHW6JCdEQ2-cCegQIABAA&oq=Step+by+steoExport+the+Object+Adobe+Photoshop+-+wikihow&gs_lcp=CgNpbWcQA1DaF1jdPGCFQ2gAcAB4AIABqgKIAYQNkgEDMi03mAEAoAEBqgELZ3dzLXdpei1pbWfAAQE&sclient=img&ei=9MQXZY2tOaCfg8UP7pKmiA0&bih=629&biw=1366&hl=en

https://www.google.com/search?q=XAMPP%2C+WAMP%2C+MAMP%2C+and+LAMP&sca_esv=569660528&hl=en&tbm=isch&source=hp&biw=1366&bih=629&ei=I7wXZc-IFfbi2roPyK2AwAI&iflsig=AO6bgOgAAAAAZRfKMy7gVqgROlsU4in5HyWY6hDrV-nv&ved=0ahUKEwiPponU1tGBAxV2sVYBHcgWACgQ4dUDCAc&uact=5&oq=XAMPP%2C+WAMP%2C+MAMP%2C+and+LAMP&gs_lp=EgNpbWciG1hBTVBQLCBXQU1QLCBNQU1QLCBhbmQgTEFNUEiBElCcBVjpD3ABeACQAQCYAa4BoAHMAqoBAzAuMrgBA8gBAPgBAvgBAYoCC2d3cy13aXotaW1nqAIA&sclient=img#imgrc=NhjcFF8drA0YHM

https://www.google.com/imghp?hl=en&tab=ri&ogbl

Development of CBLM:

The Competency Based Learning Material (CBLM) of '**Work with Basic JavaScript & jQuery' (**Occupation: Web Design, 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 Tuj Johura Arzu	Editor	01912 464 747
3	Md. Amir Hossain	Co-Ordinator	01631 670 445
4	Md. Saif Uddin	Reviewer	01723 004 419