Unit 4 : Browser Data

4.2 Browser – opening a window, giving the new window focus, window position, changing the content of window, closing a window, scrolling a web page, multiple windows at once, creating a web page in new window, JavaScript in URLs, JavaScript security, Timers, Browser location and history.

1.Define Browser . (2marks)

- A web browser is a software tool that helps you access and view content on the internet.
- This content can include web pages, images, videos, and other media, all of which are identified by URLs (web addresses).
- By clicking on hyperlinks, you can easily move between related websites or pieces of content.
- While browsers are mostly used to browse the internet, they can also be used to access information from private networks or your computer's files.
- Some of the most popular web browsers are Firefox, Google Chrome, Internet Explorer, Opera, and Safari.

2.List the operations perform on Browser. (2marks)

- We can perform various operation with browser window in JavaScript.
- Following are the operations
  - ✓ Opening a window
  - ✓ Giving new window focus
  - ✓ Window position
  - ✓ Changing content of widow
  - ✓ Closing the window
  - ✓ Scrolling window
  - ✓ Multiple window

3.Explain open () method of window object with syntax and example.(4marks)

The open() method of window object is used to open a new window and loads the document specified by a given URL.

The window.open() method is used to open a new browser window or tab and load the content from a specific URL into it. It returns a reference to the new window, which can be stored in a variable and controlled by your JavaScript.

syntax

window.open(url,name,style);

url-it specifies the URL of the new page going to be open in new window.

Name-it is used to set the name of the window.it is optional.

Style-style of the window has various parameters such as scrollbar,toolbar,height,width,location,etc, it is optional.

```
<html>
<head>
  <title>Open New Window</title>
  <script>
    function OpenNewWindow() {
      window.open('rollno.html', 'myAdWin', 'height=250, width=250');
    }
  </script>
</head>
<body>
  <button onclick="OpenNewWindow()">Open Window</button>
</body>
</html>
```

4.Giving the new Window Focus using focus() .. (4marks)

The focus() method is used to bring a window to the front and give it focus, making it the active window. When a window is opened using window.open(), the new window may not always automatically gain focus, especially if the user has multiple windows or tabs open. To ensure that the new window becomes active, you can use focus() on the window reference returned by window.open().

```
<html>
<head>
  <title>Open New Window with Focus</title>
  <script>
    function OpenNewWindow() {
      // Open a new window and store the reference
      let myWindow = window.open('rollno.html', 'myAdWin', 'height=250,
width=250');

      // Bring the new window into focus
      myWindow.focus();
    }
  </script>
</head>
<body
  <button onclick="OpenNewWindow()">Open Window with Focus</button>
</body>
</html>
```

5.**Window Position How do you control the position of a new window on the screen using window.open()? (4marks)**

To control the position of a new window on the screen when using window.open(), you can specify the top and left properties in the third parameter string. These properties determine the position of the new window relative to the top-left corner of the screen.

Positioning Parameters:

1. top: Specifies the distance in pixels from the top edge of the screen.
2. left: Specifies the distance in pixels from the left edge of the screen.

```
<html>
<head>
  <title>Open New Window with Position</title>
  <script>
    function OpenNewWindow() {
      // Open a new window with specified size and position (100px from top and 200px
from left)
      let myWindow = window.open('rollno.html', 'myAdWin', 'height=250, width=250,
top=0, left=220');
      myWindow.focus(); // Focus the new window
    }
  </script>
</head>
<body>
  <button onclick="OpenNewWindow()">Open Window with Position</button>
</body>
</html>
```

6. Changing the Content of a New Window . (4 marks)

To change the content of a newly opened window, you can access and modify its document using the document.write() method or manipulate the DOM within that window. Once you have a reference to the new window (using window.open()), you can dynamically insert HTML, text, or scripts.

```
<html>
<head>
  <title>Open New Window and Change Content</title>
  <script>
    function OpenAndChangeWindow() {
      // Open a new window
      let myWindow = window.open('', 'myNewWin', 'width=400, height=300');

      // Change the content of the new window
      var a=prompt("enter text ")
      myWindow.document.write('<h1>This is the new content</h1>');
      myWindow.document.write('<p>Here is a dynamically added paragraph.</p>');
      myWindow.document.write('<button onclick="window.close()">Close this
window</button>');

      // Focus the new window
      myWindow.focus();
    }
  </script>
</head>
<body>
  <button onclick="OpenAndChangeWindow()">Open Window and Change Content</button>
</body>
</html>
```

7.Closing the window using close() .(4marks)

If you want to focus solely on closing a window that has been opened, here's a simple example that shows just how to close an already opened window using the close() method.

```
<html>
<head>
  <title>Open New Window</title>
  <script>
    function closeWindow() {
      window.close();
    }
  </script>
</head>
<body>
  <button onclick="closeWindow()">Close Window</button>
</body>
</html>
```

8. Scrolling a Webpage in JavaScript. (4marks)

In web development, you can control the scrolling behavior of a webpage using JavaScript. Two common methods to achieve this are **scrollTo()** and **scrollBy()**.

1. scrollTo() Function

- **What It Does**: The scrollTo() function allows you to scroll the webpage directly to a specific position on the page by specifying the exact X (horizontal) and Y (vertical) coordinates.
- **Syntax**:

  window.scrollTo(xpos, ypos);

  - **xpos**: The horizontal position you want to scroll to (in pixels).
  - **ypos**: The vertical position you want to scroll to (in pixels).

2. scrollBy() Function

- **What It Does**: The scrollBy() function scrolls the webpage relative to its **current** position. You specify how many pixels to scroll by, both horizontally and vertically.
- **Syntax**:

window.scrollBy(xnum, ynum);

  - **xnum**: The number of pixels to scroll horizontally from the current position.
  - **ynum**: The number of pixels to scroll vertically from the current position.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Scroll Example</title>
</head>
<body>
    <h1>Scroll Example</h1>

    <!-- Button to scroll directly to 1500px from the top -->
    <button onclick="scrollToPosition()">Scroll to 1500px from top</button>

    <!-- Button to scroll down by 300px -->
    <button onclick="scrollByAmount()">Scroll Down by 300px</button>

    <!-- Large content to make the page scrollable -->
    <div style="height: 2000px;">
        <p id="scrollTarget" style="margin-top: 1500px;">You have scrolled to 1500px down!</p>
    </div>
```

```
•
•       <script>
•           // Function to scroll directly to 1500px from the top
•           function scrollToPosition() {
•               window.scrollTo(0, 1500);
•           }
•
•           // Function to scroll down by 300px from the current position
•           function scrollByAmount() {
•               window.scrollBy(0, 300);
•           }
•       </script>
•   </body>
•   </html>
```

9.Multiple Window Opening .

- Opening Multiple Windows: We can open multiple windows by using the window.open() function multiple times.
- Using a for loop: A for loop can be used to open multiple windows efficiently.
- Focus on the Last Window: The last window that is opened will automatically get the focus by default

```
<html>
<head>
<script type="text/javascript">
function popUp() {
    for (var i = 0; i < 5; i++) {
        // Open a new window with a unique name for each iteration
        window.open("", "MYWindow" + i, "width=300,height=300");

    }
}
</script>
</head>
<body>
    <form>
        <input type="button" value="Open Windows" onclick="popUp()">
    </form>
</body>
</html>
```

10.Creating Webpage in New Window.

To create a webpage in a new window, you can use the window.open() method in JavaScript. This method opens a new browser window or tab and can load a new webpage, show some content, or remain empty depending on how it's used.

Here's how to create a simple webpage and open it in a new window:

```
<!DOCTYPE html>
<html>
<head>
    <title>Open New Window Example</title>
    <script type="text/javascript">
        function openNewWindow() {
            // Define the HTML content for the new window
            var newWindowContent = `
                <html>
                <head>
                    <title>New Window</title>
                </head>
                <body>
                    <h1>Welcome to the new window!</h1>
                    <p>This is a simple webpage opened in a new window.</p>
                </body>
                </html>
            `;

            // Open a new window and write the content
            var newWindow = window.open("", "NewWindow", "width=400,height=400");
            newWindow.document.write(newWindowContent);
        }
    </script>
</head>
<body>
    <h1>Main Webpage</h1>
    <button onclick="openNewWindow()">Open New Window</button>
</body>
</html>
```

11. Write a javascript syntax to accessing elements of another child window.(2m)

To access elements of a child window from the parent window in JavaScript, you need to use a reference to the child window. This is usually done by storing the result of window.open() in a variable

```
// Open a child window and store the reference
var childWindow = window.open("child.html", "ChildWindow", "width=400,height=400");
childWindow.onload = function() {
  // Accessing an element with id "elementID" from the child window
  var element = childWindow.document.getElementById("elementID");
  console.log(element); // Perform actions on the accessed element
};
```

12.

Ans	Window Object Properties	
		properties of a window object-1M

Property	Description
Document	It returns the document object for the window (DOM).
Frames	It returns an array of all the frames including iframes in the current window.
Closed	It returns the Boolean value indicating whether a window has been closed or not.
History	It returns the history object for the window.
innerHeight	It sets or returns the inner height of a window's content area.
innerWidth	It sets or returns the inner width of a window's content area.
Length	It returns the number of frames in a window.
Location	It returns the location object for the window.
Name	It sets or returns the name of a window.
Navigator	It returns the navigator object for the window.
Opener	It returns a reference to the window that created the window.
outerHeight	It sets or returns the outer height of a window, including toolbars/scrollbars.
outerWidth	It sets or returns the outer width of a window, including toolbars/scrollbars
Parent	

calling onload ()-1M

calling onunload()-1M

valid function definition-1M

13. Explain the term JavaScript URL(2M)

A URL (Uniform Resource Locator) is the address of a unique resource on the internet. It is one of the key mechanisms used by browsers to retrieve published resources, such as HTML pages, CSS documents, images, and so on.

Examples of URLs:

https://developer.mozilla.org/en-US/docs/Learn/

https://developer.mozilla.org/en-US/search?q=URL

Use window.location.href to get the current URL address:

```
<!DOCTYPE html>
<html>
<body>
<h2>Get the current URL</h2>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML =
"The full URL of this page is:<br>" + window.location.href;
</script>
</body>
</html>
```

14.Define Javascript Security-

- XSS (Cross-Site Scripting): Stops hackers from adding bad scripts. Fix it by cleaning user input and using security settings like CSP.
- CSRF (Cross-Site Request Forgery): Protects against fake actions. Use special tokens and secure cookies to keep it safe.

15.Javscript timers

Or Give syntax of and explain the use of SetTime Out( ) function with the help of suitable.(4m)

Timers in JavaScript

Timers allow you to schedule tasks to happen after a delay or at regular intervals. In some cases, you might need to trigger events dynamically, such as delaying a message or performing a repeated task automatically. JavaScript provides two key functions for handling timers:

1. **setTimeout()**

- **Purpose**: The setTimeout() method is used to run a block of code **once** after a specific amount of time (in milliseconds).

Syntax: setTimeout(function, milliseconds);

Parameters:

- **function**: The function that contains the code to be executed after the delay.
- **milliseconds**: The time (in milliseconds) after which the function will be executed.

Return Value:

- The setTimeout() method returns an interval ID, which is a unique number representing the timer.

```
<!DOCTYPE html>
<html>
<head>
    <title>setTimeout Example</title>
</head>
<body>

    <script>
        function greet() {
            document.write('Hello world');  // This will be displayed after 3
seconds
        }

        setTimeout(greet, 3000);  // Delay of 3 seconds (3000 milliseconds)

        document.write('This message is shown first');  // This is displayed
immediately
    </script>

</body>
</html>
```

2.setInterval()

- **Purpose**: The setInterval() method is used to repeatedly execute a block of code **at specified time intervals** (in milliseconds).
- **Syntax**: setInterval(function, milliseconds);

Parameters:

- **function**: The function that contains the code to be executed repeatedly.
- **milliseconds**: The time (in milliseconds) between each execution of the function.

Return Value:

- The setInterval() method returns an interval ID, which is a unique number that represents the interval.

```
<!DOCTYPE html>
<html>
<head>
    <title>setInterval Example</title>
</head>
<body>

    <script>
        function greet() {
            document.write('Hello world<br>');  // This message will be displayed every 2 seconds
        }

        setInterval(greet, 2000);  // Repeat every 2 seconds (2000 milliseconds)

        document.write('This message is shown first<br>');  // This is displayed immediately
    </script>

</body>
</html>
```

Difference Between `setTimeout()` and `setInterval()`

Feature	`setTimeout()`	`setInterval()`
Purpose	Executes a function **once** after a specified delay.	Executes a function **repeatedly** at fixed intervals.
Execution	Runs the function after a specified time, only **once**.	Runs the function repeatedly after each time interval until stopped.
Use case	Delaying a task, such as showing a message after a few seconds.	Repeating tasks, like updating a clock every second.
Syntax	`setTimeout(function, milliseconds)`	`setInterval(function, milliseconds)`
Stop the Function	Can be stopped using `clearTimeout(timeoutID)`.	Can be stopped using `clearInterval(intervalID)`.

1. Window Location Object

The window.location object is an essential part of the web browser's API that provides information about the current URL and allows manipulation of the URL. Here are the key properties and methods:

Properties

- **window.location.href**:
    - Returns the complete URL of the current page.
    - Example: If you're on https://example.com/page1, window.location.href would return that entire URL.
- **window.location.hostname**:
    - Returns the domain name of the web server without the protocol or path.
    - Example: For https://example.com/page1, it returns example.com.
- **window.location.pathname**:
    - Returns the path and filename of the current page.
    - Example: For https://example.com/page1, it returns /page1.
- **window.location.protocol**:
    - Returns the protocol used (e.g., http: or https:) of the current page.
    - Example: For https://example.com/page1, it returns https:.

Methods

- **window.location.assign(url)**:
    - Loads the specified URL, replacing the current page in the browser history.
    - Example: window.location.assign("https://example.com/page2") will navigate to page2.
- **window.location.reload(forceReload)**:
    - Reloads the current page. If forceReload is set to true, it will reload from the server instead of cache.
    - Example: window.location.reload(true) forces a reload from the server.

Brower History :The window.history object provides access to the browser's session history (the pages visited in the current tab or window). It is important to note that JavaScript has limited access to this object to protect user privacy.

- The ***window.history*** object contains the record of URL's visited by the user within a browser window.
- The History object is a part of the window object so that it is accessed by ***window.history.***
- **Properties**
  - ✓ ***length*****-** it is used to return the number of URL's in the list of history.
- **Methods**
  - ✓ **back()-**it is used to load the previous URL in the history list.
  - ✓ **forward()-**it is used to load the next URL in the history list.
  - ✓ **go()-** it is used to load the specific URL from the history list.

Mauli IT

```
<!DOCTYPE html>
<html>
<head>
  <title>History Example</title>
  <script>
    function goBack() {
      window.history.back(); // Navigates to the previous page
    }
  </script>
</head>
<body>
  <input type="button" value="Back" onclick="goBack()">
</body>
</html>
```

Program 1 : Write HTML code to design a form that displays two buttons START and STOP. Write a JavaScript code such that when user clicks on START button, real time digital clock will be displayed on screen. When user clicks on STOP button, clock will stop displaying time. (Use Timer methods) – 6 marks

```
<html>
<head>
<script type="text/javascript">
    var clock;

    function updateClock() {
        var d = new Date();
        var hours = d.getHours();
        var minutes = d.getMinutes();
        var seconds = d.getSeconds();
        var currentTimer = hours + " : " + minutes + " : " + seconds;
        document.getElementById("timer").innerHTML = currentTimer;
    }

    function StartTimer() {
        updateClock(); // Show time immediately
        clock = setInterval(updateClock, 1000); // Update every second
    }

    function StopTimer() {
        clearInterval(clock); // Stop the timer
    }
</script>
</head>
<body>
    <form>
        <input type="button" value="START" onclick="StartTimer()">
        <input type="button" value="STOP" onclick="StopTimer()">
        <p id="timer"></p>
    </form>
</body>
</html>
```

Program 2 : Write a javascript program to changing the contents of a window.(2m)

```
<head><title>Change Content</title>
</head>
<body>
 <p id="content">Original Content</p>
    <button onclick="changeContent()">Change Content</button>

    <script>
        function changeContent() {
            document.getElementById("content").innerHTML = "New Content
Updated!";
        }
    </script>
```

Program 3: Write a javascript to open a new window and the new window is having two frames. One frame containing buthon as “click here !”, and after clicking this button an image should open in the second frame of that child window.(6marks)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Open New Window with Frames</title>
    <script>
        function openNewWindow() {
            // Open a new window
            var newWin = window.open("", "", "width=600,height=400");

            // Write the HTML structure for two frames in the new window
            newWin.document.write(`
                <html>
                <head>
                    <title>Two Frames</title>
                </head>
                <frameset cols="50%,50%">
                    <frame name="buttonFrame">
                    <frame name="imageFrame">
                </frameset>
                </html>
            `);

            // Write the button into the first frame (buttonFrame)
            newWin.frames["buttonFrame"].document.write(`
                <html>
                <body>
                    <button
onclick="parent.frames['imageFrame'].location.href='images.jpg';">Click Here!</button>
                </body>
                </html>
            `);
        }
    </script>
</head>
<body>
    <button onclick="openNewWindow()">Open New Window with Frames</button>
</body>
</html>
```

Program 4: Write a JavaScript function that will open new window when the user will clicks on the button. (4marks)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Open New Window with Frames</title>
    <script>
        function openNewWindow() {
          window.open("timer.html","OpewindowExmaple","height=600 ,
width=600");
        }
    </script>
</head>
<body>
    <button onclick="openNewWindow()">Open New Window with Frames</button>
</body>
</html>
```

Program 5 : Write a JavaScript that accepts user's first name and domain name of organization from user. The JavaScript then forms email address as and displays the results in the browser window. (4marks)

```
<!doctype html>
<html>
    <head>
        <title>Genrating Email address</title>
    </head>
    <body>
        <form>
            <label>Enter User's First Name :</label>
            <input type="text" name="firstname" id="name" placeholder="eg.
rahul"><br><br>
            <label>Enter Domain name of Organization :</label>
            <input type="text" name="domain" id="domain" placeholder="eg.
sjcem.edu.in"><br>
            <input type="button" onclick="abc()" value="Genrate Email">
            <br><br>
            <p id="output">email id here:</p>
        </form>
```

```
        <script>
            function abc()
            {
                var s1=document.getElementById('name').value;
                var s2=document.getElementById('domain').value;
                var s3=s1+"@"+s2;
                document.getElementById('output').innerHTML=s3;
            }
        </script>
    </body>
    </html>
```

Program 6 : Write a JavaScript that opens a new popup window with message "WELCOME To SCRIPTING" when the page loads and a new popup window displaying message "FUN WITH SCRIPTING" when the page unloads. (6marks)

```
<!doctype html>
<html>
    <head>
        <title>Page load and Unload</title>
    </head>
    <body onload="openLoad()" unload="closeLoad()">
        <h3> check load and unload functionality </h3>
        <script>
            function openLoad()
            {
                var win=window.open("","openwindow","height=500,width=500");
                win.document.write("WELCOME To SCRIPTING");
            }
            function closeLoad()
            {
                var win=window.open("","closewindow","height=500,width=500");
                win.document.write("FUN WITH SCRIPTING");
            }
        </script>
    </body>
</html>
```

Question

1. **Write a JavaScript program that uses the setInterval() function to change the background color of a webpage every second.**
2. **Describe the use of the history object in JavaScript. Write a JavaScript program to navigate forward and backward using the browser's history.**
3. **Write a JavaScript program that opens a popup window and closes it automatically after 5 seconds.**
4. **Write a JavaScript function that returns the factorial of a number using recursion.**
5. **Write a JavaScript program to display a countdown timer that starts at 10 and decreases by 1 every second, and displays "Time's up!" when the countdown reaches 0.**