## Assignment 5. Advanced JavaScript and DOM Manipulation

### Objective

In this assignment, students will create an interactive web application centered around their project's theme. The project should showcase JavaScript fundamentals, DOM manipulation, event handling, and animations, focusing on engagement and user-friendliness.

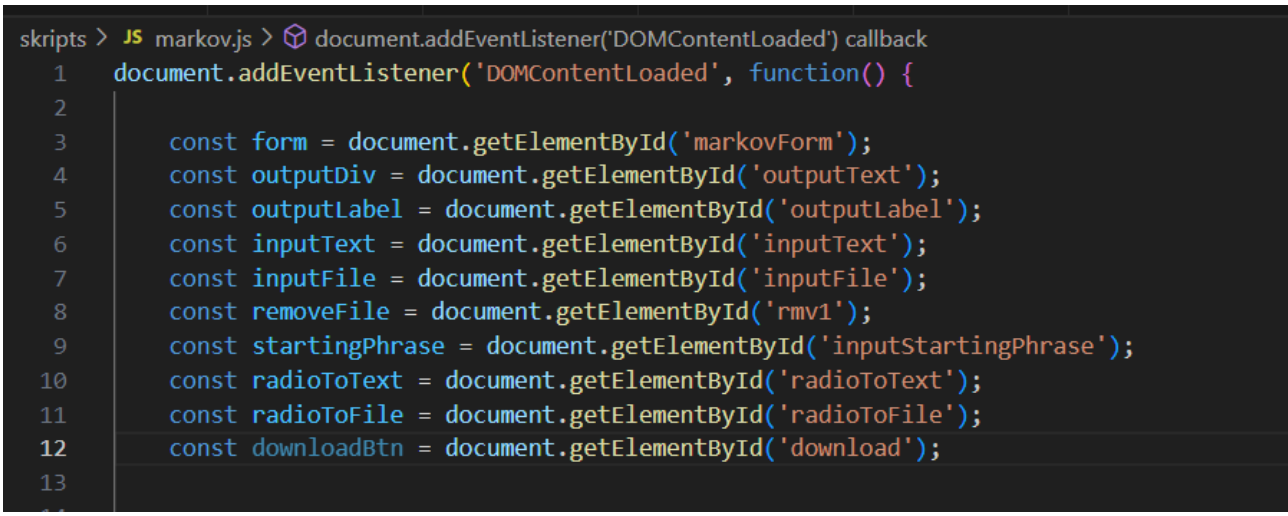
---

### Requirements

#### 1. DOM Manipulation and Styling

**Selecting and Manipulating HTML Elements**

- Use `document.querySelector` or similar methods (e.g., `getElementById`, `getElementsByClassName`, `getElementsByTagName`) to select HTML elements. Demonstrate the ability to target specific elements effectively, whether they are unique or part of a group. For example, use `document.querySelectorAll` to select rating stars (e.g., icons). Allow users to click on a star to set their rating, changing its color to indicate selection.




- Modify the content of selected elements dynamically using properties such as `textContent`, `innerHTML`, `innerText`. This may involve updating text

based on user interactions or external data sources. For example, change a message displayed on the page when a button is clicked or when an event occurs.

```
function updateResults(results) {
    document.getElementById('wordCount').textContent = results.wordCount;
    document.getElementById('charCountWithSpaces').textContent = results.charCountWithSpaces;
    document.getElementById('charCountWithoutSpaces').textContent = results.charCountWithoutSpaces;
    document.getElementById('lineCount').textContent = results.lineCount;
    document.getElementById('sentenceCount').textContent = results.sentenceCount;
    document.getElementById('paragraphCount').textContent = results.paragraphCount;
    document.getElementById('avgWordsPerSentence').textContent = results.avgWordsPerSentence;
    document.getElementById('avgCharsPerWord').textContent = results.avgCharsPerWord;
}
```

**Dynamic Style Changes**

- Implement methods to modify styles directly through JavaScript. This can include changing CSS properties such as background color, font size, visibility, and positioning. Use properties like `style.backgroundColor`, `style.display`, `style.transform` to create a dynamic user interface.

```
function changeBg() {
    // toggle interval
    if (bgInterval) {
        clearInterval(bgInterval);
        bgInterval = null;
        document.getElementById("canvas1").style.background = ``
    } else {
        bgInterval = setInterval(() => {
            document.getElementById("canvas1").style.background = `hsl(${Date.now()/10%360}, 50%,50%)`
        });
    }
}
```

OR

- Create a functional switch to toggle between Day and Night themes. This could involve changing colors, backgrounds, and text styles. Use classes or inline styles to achieve the desired effect and enhance user experience.

**Manipulating Attributes (examples below)**

Implement a “Read More” button that, when clicked, toggles the visibility of additional content by changing the `style.display` property.

OR

```
let greetingMsg = `Good day, my fella ${data.name}`
switch ( Math.floor(now / 6) ) {
    case 0: greetingMsg =  `Not sleeping, <span class="text-primary">${data.name}</span>?` // 0-5
    case 1: greetingMsg =  `Wakey, wakey, <span class="text-primary">${data.name}</span>!` // 6-11
    case 2: greetingMsg =  `Good day my fella <span class="text-primary">${data.name}</span>?` // 11-17
    case 3: greetingMsg =  `Isn't it a good evening, <span class="text-primary">${data.name}</span>?` // 18-23

}
contactUsForm.querySelector('#greeting').innerHTML = greetingMsg;
```

**2. Create an image gallery with thumbnails. When a thumbnail is clicked, dynamically change the `src` of a larger display image to show the selected thumbnail.**

OR

- Use `document.getElementById` to select a greeting elem ent. Create an input field that allows users to enter their name, which updates the greeting dynamically when they submit.

OR

- Use `document.querySelectorAll` to select rating stars (e.g., icons). Allow users to click on a star to set their rating, changing its color to indicate selection.

OR

- Use `document.querySelector` to select a content area. Implement a button that, when clicked, fetches and displays new content (e.g., a random fact or quote) in that area.

**3. Event Handling**

- **Event Listeners on Buttons**
  Implement at least one button that triggers an action when clicked using addEventListener. Examples:
  - Create a button that, when clicked, displays the current time in a designated area. Use `new Date().toLocaleTimeString()` in the button's click event to get the current time.

  OR

  - Make a reset button that clears all inputs in a form when clicked.Use `document.querySelectorAll('input').forEach(input => input.value = '')` to reset form fields.

  OR

- Create a button that loads additional content (like more posts or products) without refreshing the page. Use `fetch` to retrieve content from a server or an API and append it to a specific container.

- **Keyboard Event Handling**

  Use a keydown or keypress event to detect key presses and trigger actions.

  - Implement keyboard navigation for a navigation menu so users can use arrow keys to move between items. Capture `keydown` events and adjust focus to the next/previous menu item based on arrow key presses.

```
    document.addEventListener('keydown', function(event) {
        let linkId = links.indexOf(new URL(window.location.href).pathname);
        if (linkId < 0) {
            return
        }
        if (event.key === 'ArrowRight') {
            linkId = (linkId + 1) % links.length;
            window.location.href = links[linkId];
        } else if (event.key === 'ArrowLeft' ) {
            linkId = (linkId - 1 + links.length) % links.length;
            window.location.href = links[linkId];
        }
    })
})
```

- **Responding to Events with Callbacks**

  Implement a callback function to handle user interactions.

  - Create a contact form that submits data asynchronously and displays a success message without page reload. Use `fetch` to POST the form data and provide feedback through a callback function.

```
// Callback function to handle form submission
async function handleFormSubmit(event) {
    event.preventDefault(); // Prevent the default form submission

    const formData = new FormData(contactUsForm);
    const data = Object.fromEntries(formData.entries());

    try {
        const response = await fetch('/api/contact', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(data),
        });

        if (response.ok) {
            const result = await response.json();
            document.querySelector('.fw-bold.text-center').innerText = 'Your message has been sent successfully!';
            contactUsForm.reset(); // Reset the form fields
        } else {
            throw new Error('Network response was not ok.');
        }
    } catch (error) {
        document.querySelector('.fw-bold.text-center.text-danger').innerText = 'There was an error sending your message. Please try again.';
        console.log(error)
    }
}

contactUsForm.addEventListener('submit', handleFormSubmit);
```

OR

- Implement a multi-step form where users can navigate between steps using “Next” and “Back” buttons. Use a callback function to display the correct step based on user actions, maintaining the state of inputs.

- **Switch Statements**

  Use a switch statement to control logic based on user input or interaction (e.g., responding differently to various key presses or button clicks).

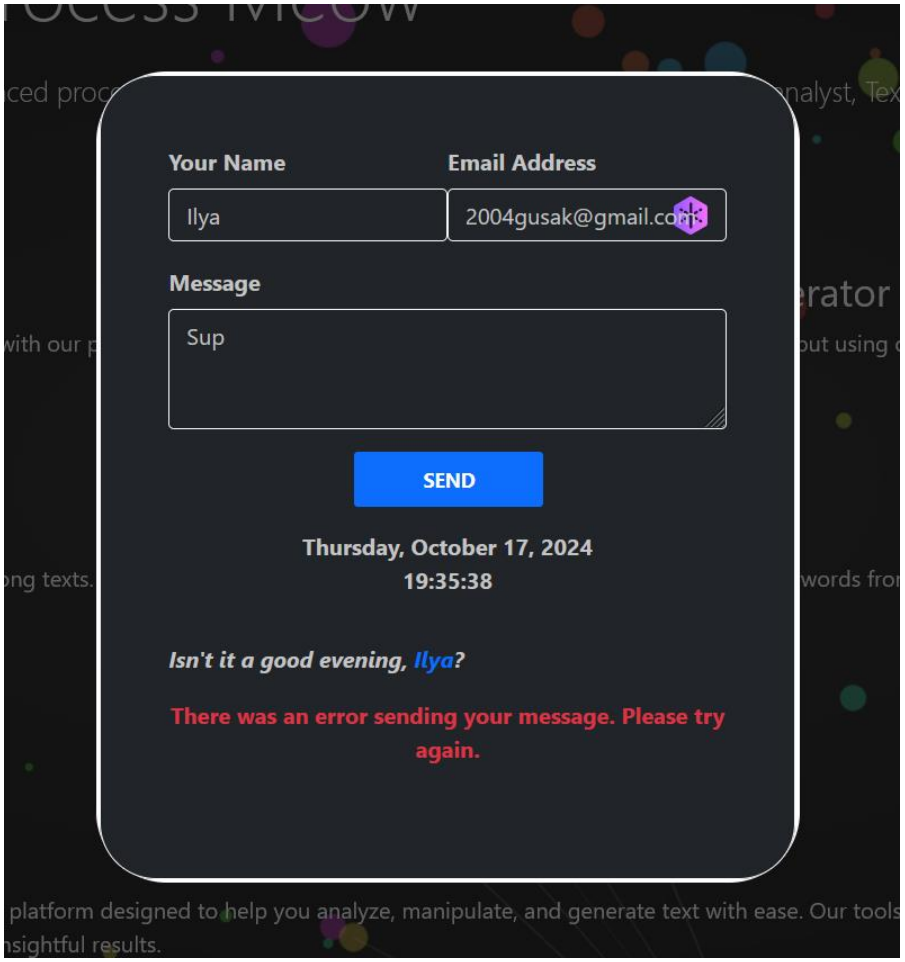
  - Build a product filtering system (e.g., e-commerce site) where users can filter products by category or price range.

  OR

  - Create a news feed application that allows users to switch between different categories (e.g., Sports, Technology) using buttons.

  OR

  - Create a function that displays a different greeting based on the time of day (morning, afternoon, evening).




OR

- Implement a language selector that changes the language of a website based on user choice (e.g., English, Russian, Kazakh).

## 4. JavaScript Advanced Concepts

- **Objects and Methods**
  - Use JavaScript objects and their methods to structure data or handle logic in the project. Display or manipulate object properties on the page.

```
function buildMarkovChain(words, order) {
    const markovChain = {};
    for (let i = 0; i < words.length - order; i++) {
        const currentState = words.slice(i, i + order).join(' ');
        const nextWord = words[i + order];

        if (!markovChain[currentState]) {
            markovChain[currentState] = {};
        }

        if (!markovChain[currentState][nextWord]) {
            markovChain[currentState][nextWord] = 0;
        }

        markovChain[currentState][nextWord]++;
    }
    return markovChain;
}
```

- **Arrays and Loops**
  - Use arrays to manage collections of items and loops (for, while) to iterate over elements (e.g., to display a list of items dynamically).
- **Higher-Order Functions**
  - Implement at least one higher-order function that takes another function as an argument (e.g., using map, filter, forEach).

```
navLinks.forEach((a) => {
    if (a.href.length > 0 && window.location.href.length > 0) {
        links.push(new URL(a.href).pathname)
        if (new URL(a.href).pathname === new URL(window.location.href).pathname) {
            a.classList.add('text-primary')
        }
    }
})
```

- **Play Sounds**
  - Use JavaScript to trigger sound effects. Example – set up a button that plays a notification sound when clicked.

```
if(!audio.paused && !audio.ended) {
    audio.pause();
}
else if (audio.paused) {
    audio.play();
}
```

- **Animations**

- Add animations to elements (e.g., using style.transform or CSS transitions), triggered by events.

```
function toggleAnswer(answers) {
    for (var i = 0; i < answers.length; i++) {
        var child = answers[i];
        child.classList.toggle('collapsed');
    }
}
```

```
.faq-answer {
    padding-left: 4%; /* some padding on the left */
    color: #fff; /* darker gray color for answers */
    transition: opacity .5s, max-height .3s; /* adjust the duration as needed */
    overflow: hidden;
    max-height: 500px; /* set a maximum height for the transition */
}
.collapsed {
    opacity: 0;
    max-height: 0;
    border: 0;
    margin: 0;
}
```

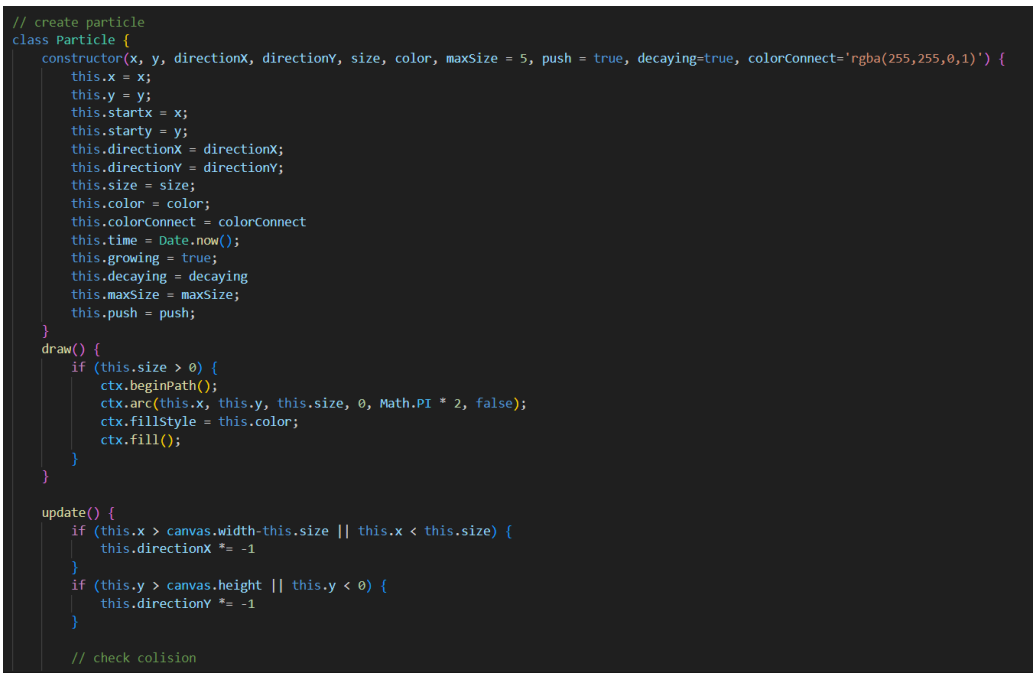
## 5. Structure and Separation of Concerns

- **Separation of HTML, CSS, and JavaScript**
  - Ensure that the structure (HTML), style (CSS), and behavior (JavaScript) are clearly separated. Avoid inline styling or scripts directly in HTML elements.

```
        </footer>

        <canvas id="canvas1" style="z-index: -1;"></canvas>
        <script src="skripts/QuadTree.js"></script>
        <script src="skripts/particle.js"></script>
        <script src="skripts/counter.js"></script>
    </div>
</body>
</html>
```

- **Clean Code and Readability**
  - Write well-organized, readable code. Use comments where necessary to explain logic.



```
// create particle
class Particle {
    constructor(x, y, directionX, directionY, size, color, maxSize = 5, push = true, decaying=true, colorConnect='rgba(255,255,0,1)') {
        this.x = x;
        this.y = y;
        this.startx = x;
        this.starty = y;
        this.directionX = directionX;
        this.directionY = directionY;
        this.size = size;
        this.color = color;
        this.colorConnect = colorConnect
        this.time = Date.now();
        this.growing = true;
        this.decaying = decaying
        this.maxSize = maxSize;
        this.push = push;
    }
    draw() {
        if (this.size > 0) {
            ctx.beginPath();
            ctx.arc(this.x, this.y, this.size, 0, Math.PI * 2, false);
            ctx.fillStyle = this.color;
            ctx.fill();
        }
    }

    update() {
        if (this.x > canvas.width-this.size || this.x < this.size) {
            this.directionX *= -1
        }
        if (this.y > canvas.height || this.y < 0) {
            this.directionY *= -1
        }

        // check colision
```

- **Proper Use of Functions and Variables**
  - Use functions to encapsulate repeated code, and choose meaningful variable names.

## 6. Creativity and Engagement

- **Theme Integration**
  - The project should reflect your team's theme and be relevant to it. Make the content engaging and creative in a way that aligns with your theme.
- **User Experience**
  - Ensure the project is easy to use and intuitive for the user. ●

**Fun and Interactivity**

  - Add fun or engaging elements, such as sounds, animations, or interactive content that encourage user engagement.

---

## Submission Instructions

1. Upload your project files (HTML, CSS, and JavaScript) in a zip folder.
   a. All team members must upload project files to the moodle.
2. Prepare to describe during defense how each feature from the requirements was implemented. Reflect on challenges encountered and how they were solved.

## Grading Criteria

| Criteria | Description | Weight |
|---|---|---|
| **DOM Manipulation and Styling** | Selecting elements, modifying content, changing styles, and manipulating attributes. | 10% |
| **Event Handling** | Implementing event listeners, keyboard handling, callbacks, and switch statements for event responses. | 10% |
| **Dynamic Interaction** | Implementing drag-and-drop features or integrating external API to fetch and display data. | 5% |
| **Data Structures and Functions** | Using objects, arrays, loops, and higher-order functions to manage data and logic in the project. | 10% |
| **Sound Effects** | Triggering sound effects based on user interactions (e.g., button clicks, key presses). | 5% |
| **Animations** | Adding animations to elements using CSS transitions or JavaScript effects triggered by events. | 5% |
| **Clean Code and Structure** | Ensuring clear separation of HTML, CSS, and JavaScript, clean code, and proper function/variable usage. | 5% |
| **UX and Engagement** | Integrating the project theme, ensuring user experience, and adding engaging elements. | 10% |
| **Feature Implementation Explanation** | Clearly describe how each feature from the requirements was implemented. | 15% |

| | | |
|---|---|---|
| **Answering Theoretical Questions** | Answer theoretical questions related to the project and concepts used. | 15% |
| **Making Changes and Adjustments** | Demonstrate ability to make changes to the project based on feedback or questions. | 10% |