Help to complete the tasks of this exercise can be found on the chapter 13 ” JavaScript and the browser” ,chapter 14 “The Document Object Model” and chapter 15 “Handling events” of our course book “Eloquent JavaScript” (3rd edition) by Marijin Haverbeke. The aims of the exercise are to become familiar with DOM and how JavaScript events are handled and used.

Embed your theory answers, drawings, codes and screenshots directly into this document. Always immediately after the relevant question. Return the document into your return box in the Optima platform by the deadline.

It’s also recommendable to use Internet sources to supplement the information provided by the course book.

The maximum number of points you can earn from this exercise is 10.

Tasks:

1. HTML DOM (2 points)
    a. What is HTML DOM? (0,5 points)

    DOM= Document Object Model. It is Object Model for HTML, and API for JavaScript.

    b. What JavaScript can do to web pages by utilizing HTML DOM? (0,5 points)

    It can access and change all the elements of an HTML document.

    c. List and shortly explain at least three different kind of HTML Node types? (0,5 points)

    Element represents an element, it has children such as: Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference. Returns element name

    Attr represents an attribute, children are Text, EntityReference. Returns attribute name or attribute value.
    Document represents the entire document, children are Element, ProcessingInstruction, Comment, DocumentType. Returns #document

    d. Criticize HTML DOM. (0,5 points)

    It is “rough” to use, but it works widely with different kind of languages. When DOM was designed, it was planned to work with MANY different languages, which may have caused it to be so constraint.
    It also has cross browsers inconsistencies, but not sure what those are.

2. List and explain different ways to attach event handlers in JavaScript. What are the pros and cons of each of them? (1 point)

   Onevent,

   addEventListener(), +Ease to use, you can also remove it.

   -No IE 8 compatibility and previous

   attachEvent + IE8 compatibility and previous

3. Create an application that makes it possible to add messages on the web page. The web page displays a text area into which a new message can be written. The web page also displays a button “Add a new message!” that can be used to add a new message on the page. The message is added as a new paragraph on the page below the text area and the button when the button is clicked. The new message doesn’t replace the old ones already on the page. (2 points)

   We’ll do this together during the lesson.

Lesson 5 > Task 3 > JS EX05T3.js > ...

```
const initializeApplication = () =>{

    let addButton = document.getElementById("add");

    addButton.addEventListener("click", () => {

        let newMessage = document.getElementById("newmessage").value;

        let newPN;//Paragraph node
        let newTN;//Text Node


        if (newMessage && newMessage.length > 0) {
            newPN=document.createElement('p');
            newTN=document.createTextNode(newMessage);

            newPN.prepend(newTN)

            let allMessages=document.getElementById("messages");

            allMessages.prepend(newPN)


            document.getElementById("newmessage").value="";

        }

    });

};


window.addEventListener("load", () => {
    initializeApplication();
})
```

4. Create a page with 3 buttons. Layout is not important. When button 1 is pressed, background of a page should change to blue, button 2 changes it to gray and button 3 resets background to white (2 pts)

```
const initializeApplication = () =>{

    let button1 = document.getElementById("first");
    let button2 = document.getElementById("second");
    let button3 = document.getElementById("third");

    button1.addEventListener("mousedown", event => {
        if (event.button == 0){
            document.body.style.background="blue";
        }
    })
    button2.addEventListener("mousedown", event => {
        if (event.button == 0){
            document.body.style.background="gray";
        }
    })
    button3.addEventListener("mousedown", event => {
        if (event.button == 0){
            document.body.style.background="white";
        }
    })
};


window.addEventListener("load", () => {
    initializeApplication();
})
```

5. Related to event handlers, answer 2 out of the following 3 questions. (1 point)
    a. Why would you sometimes like to control event propagation? (0,5 points)

       Sometimes we don't want overlapping effects, with this we can cancel some events, in correct situations.
    b. Why would you sometimes like prevent a default action? (0,5 points)

       For example, to prevent link from sending you to the links page.

    c. What is debouncing an event? (0,5 points)

6. Make exercise “Balloon” from book (Exercises, chapter 15). (2 pts)

```
const initializeApplication = () =>{



    let balloon = document.getElementById("balloon");
    balloon.style.fontSize="20px";
    document.addEventListener("keydown", balloonGrow);


    function balloonGrow(event) {
    let currentSize = parseInt(balloon.style.fontSize);
    let changeAmount=5;

        if (event.key == "ArrowUp") {
            balloon.style.fontSize=(currentSize + changeAmount) + 'px';


            if ((currentSize + changeAmount) > 100){
                console.log("Kill");
                document.removeEventListener("keydown", balloonGrow);
                document.getElementById("balloon").innerHTML = "💥";
            } else {
                console.log("Not yet")
            }



        } else if (event.key == "ArrowDown") {
            balloon.style.fontSize=(currentSize-changeAmount) + 'px';
            console.log("Down");
    }
        }

    };




window.addEventListener("load", () => {
    initializeApplication();
})
```