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)

    HTML DOM (Document Object Model) on JavaScript-rajapinta, jonka avulla voidaan muokata ja hallita HTML-dokumenttia. DOM esittää HTML-dokumentin objektimallinnuksena, jossa eri HTML-elementit ovat objekteja ja niiden väliset suhteet ovat objektien välisiä suhteita.

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

    JavaScript voi HTML DOM:in avulla muokata verkkosivun sisältöä, tyyliä ja rakennetta. Esimerkiksi JavaScript voi:

    - Lisätä, poistaa ja muokata HTML-elementtejä
    - Muuttaa HTML-elementtien tyyliä
    - Vaihtaa HTML-elementtien järjestystä
    - Lisätä ja käsitellä verkkosivuilla tapahtuvia tapahtumia, kuten klikkauksia ja näppäimistön painalluksia

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

    HTML-solmuja on erilaisia tyyppejä, mutta kolme yleisintä tyyppiä ovat:

    - Elementtisolmut (element nodes): Elementtisolmut esittävät HTML-elementtejä.
    - Tekstisolmut (text nodes): Tekstisolmut esittävät elementtien välissä olevaa tekstiä.
    - Kommenttisolmut (comment nodes): Kommenttisolmut esittävät HTML-kommentteja.

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

    on tehokas ja joustava tapa muokata verkkosivuja, mutta siinä on myös joitakin puutteita. Esimerkiksi HTML DOM voi olla monimutkainen ja vaikea oppia. Lisäksi HTML DOM:in eri selaimissa on eroja, mikä voi tehdä verkkosivujen kehittämisestä haastavaa.

    Erilaisia tapoja liittää tapahtumakäsittelijöitä JavaScript-koodissa

    JavaScript-koodissa tapahtumakäsittelijöitä voidaan liittää elementteihin seuraavilla tavoilla:

    - addEventListener()-metodi: Tämä on yleisin tapa liittää tapahtumakäsittelijöitä elementteihin. addEventListener()-metodi ottaa kaksi argumenttia: tapahtuman tyypin ja tapahtumakäsittelijöfunktion.
    - onclick-, onmouseover-, onmouseout- jne. attribuutit: Näitä attribuutteja voidaan käyttää liittämään tapahtumakäsittelijöitä elementteihin HTML-koodissa. Esimerkiksi onclick-attribuutti liittää elementtiin klikkaus-tapahtumakäsittelijän.
    - Elementin tapahtuma-attribuutit (element event properties): Elementtien tapahtuma-attribuutteja voidaan käyttää liittämään tapahtumakäsittelijöitä elementteihin JavaScript-koodissa. Esimerkiksi elementin onclick-attribuutti liittää elementtiin klikkaus-tapahtumakäsittelijän.

    Eri tapahtumakäsittelijöiden edut ja haitat:

Tapahtumakäsittelijä	Edut	Haitat
addEventListener()		Monimutkaisempi oppia kuin HTML-attribuuttien käyttäminen
onclick-, onmouseover-, onmouseout- jne. attribuutit	Yksinkertaisia käyttää	Ei yhtä joustavia kuin addEventListener()
Elementtien tapahtuma-attribuutit	Samat edut ja haitat kuin onclick-, onmouseover-, onmouseout- jne. attribuuteill	

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

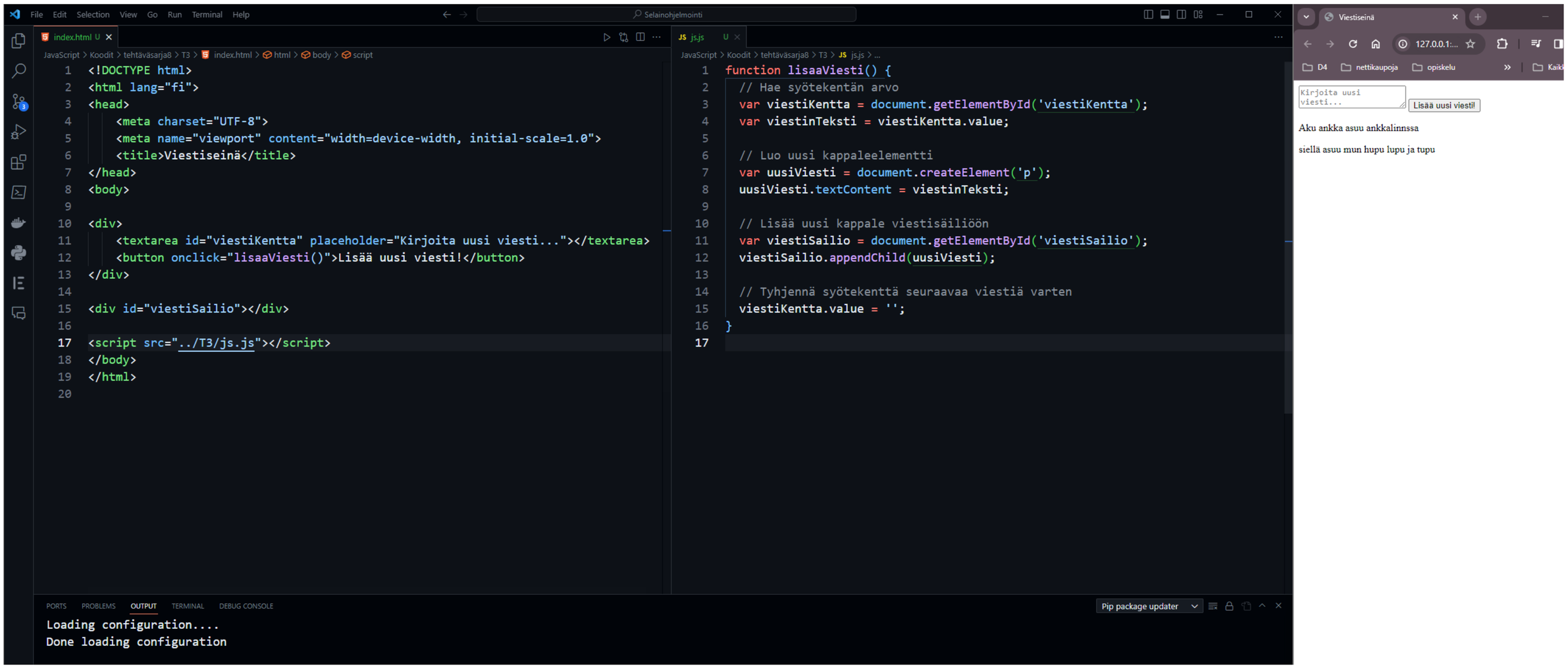
   Tapahtumankäsittelijöiden liittämisessä JavaScriptissä on erilaisia tapoja. Inline-tapahtumankäsittelijät määritellään suoraan HTML:ssä, mutta ne voivat sekoittaa HTML:n ja JavaScriptin. Perinteiset DOM-tapahtumankäsittelijät käyttävät DOM:n on-ominaisuuksia, mutta rajoittavat yhden käsittelijän tapahtumatyypille.

   DOM-tason 2 tapahtumankäsittelijät (addEventListener) mahdollistavat useiden käsittelijöiden käytön samalle tapahtumatyypille ja ovat moderni tapa. HTML:n on* -attribuutit JavaScriptin kanssa tarjoavat paremman erottelun kuin inline, mutta säilyttävät inline-tyyppisen käyttäytymisen.

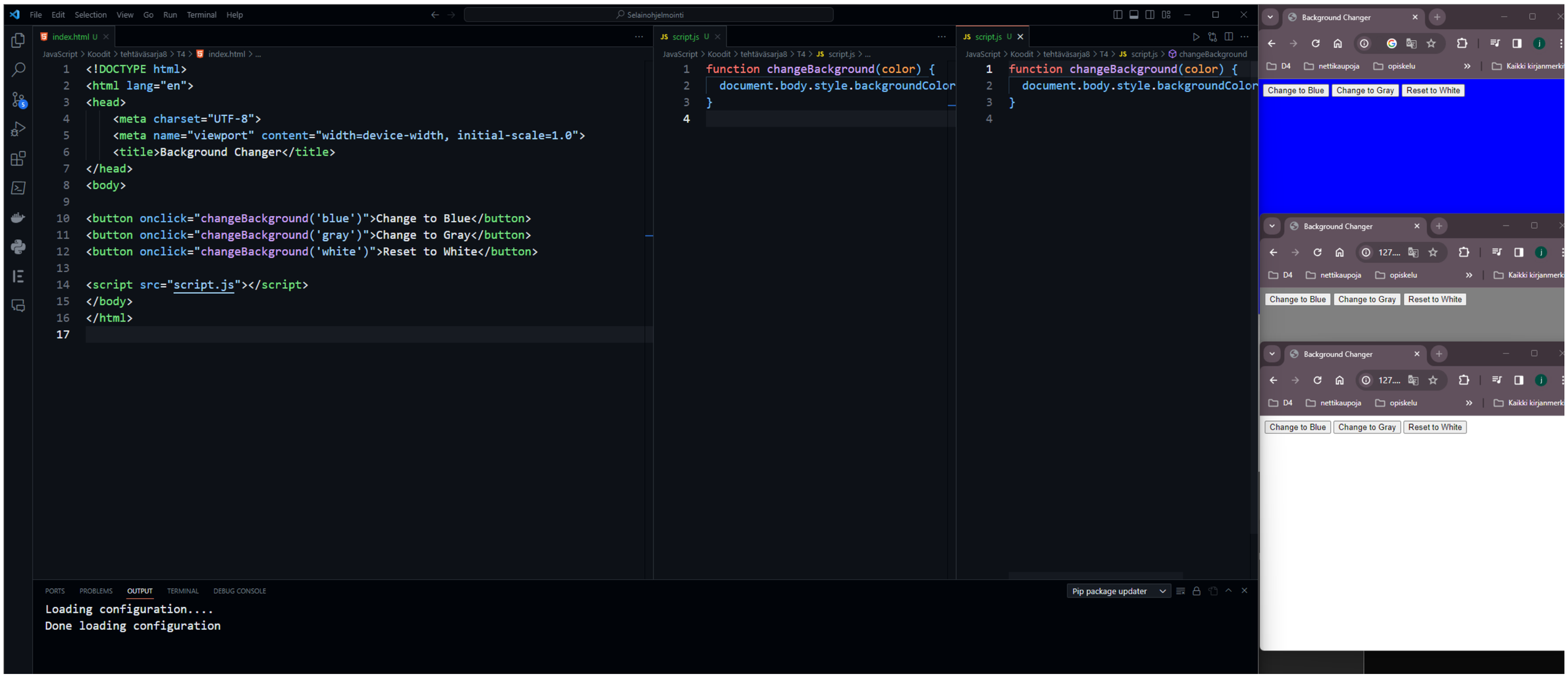
   addEventListener käyttäen this mahdollistaa pääsyn elementtiin this-avainsanalla käsittelijässä. Tapahtumien delegointi on tehokas tapa käsitellä suurta määrää elementtejä yhdellä käsittelijällä, erityisesti dynaamisessa sisällössä. Valinta riippuu projektin vaatimuksista, ylläpidettävyydestä ja koodityylistä.

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.



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)



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)

      DOMissa tapahtuvassa tapahtuman leviämisessä on kaksi vaihetta: sieppaus (capturing) ja kupliminen (bubbling). Tapahtuman leviämisen hallitseminen voi olla hyödyllistä, kun haluat hallita tapahtumankäsittelijöiden suoritusjärjestystä. Esimerkiksi voit siepata tapahtuman dokumenttitasolla ennen kuin se tavoittaa tietyn elementin tai käsitellä sitä kuplimisvaiheessa, kun se liikkuu ylöspäin DOM-puussa. Tämä mahdollistaa hienojakoisemman hallinnan siitä, miten tapahtumia käsitellään sovelluksessasi

   b. Why would you sometimes like prevent a default action? (0,5 points)

      Oletustoiminnon estäminen on hyödyllistä, kun haluat korvata tietyn HTML-elementin tapahtumaan liittyvän oletuskäyttäytymisen. Esimerkiksi voit estää lomakkeen lähettämisen, kun käyttäjä napsauttaa lähetyspainiketta, jotta voit käsitellä lomakkeen lähetyksen JavaScriptin ja AJAXin avulla. Tämä tehdään usein dynaamisempien ja vuorovaikutteisempien käyttöliittymien luomiseksi samalla kun säilytetään hallinta selaimen määrittämistä oletuskäyttäytymisistä

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

      Tapahtuman "debouncella" tarkoitetaan viivästyksen ottamista ennen kuin tietty tapahtumankäsittelijä suoritetaan. Tätä käytetään yleisesti tilanteissa, joissa tapahtuma, kuten näppäimen painallus tai vieritys, voi laukaista funktion useita kertoja nopeasti peräkkäin. Debouncen avulla varmistetaan, että liittyvää funktiota kutsutaan vain, kun tapahtuma on lakannut tapahtumasta määritellyn ajanjakson ajan. Tämä voi auttaa optimoimaan suorituskykyä, vähentämään tarpeettomia funktiokutsuja ja parantamaan kokonaiskäyttäjäkokemusta estämällä nopeita ja toistuvia toimintojen suorituksia.

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

Kuvassa olen jo suurentanut palloa niin paljon että se on räjähtänyt