**ЛАБОРАТОРНА РОБОТА №6**

**Тема:**

**КЛАСИ. ОБ'ЄКТНА МОДЕЛЬ ДОКУМЕНТА (DOM). ПОДІЇ. ОБ'ЄКТ ПОДІЇ. ДЕЛЕГУВАННЯ ПОДІЙ. THROTTLE. МОДУЛЬНОСТЬ КОДУ**

**Мета**: придбати практичні навички роботи з об'єктами. Методи об'єкта.. Callback. Стрілочні функції. Стрілочні функції як колбеки[​](https://textbook.edu.goit.global/javascript-yk5evp/v2/uk/docs/lesson-07/arrow-fns" \l "%D1%81%D1%82%D1%80%D1%96%D0%BB%D0%BE%D1%87%D0%BD%D1%96-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%97-%D1%8F%D0%BA-%D0%BA%D0%BE%D0%BB%D0%B1%D0%B5%D0%BA%D0%B8" \o "Пряме посилання на цей заголовок).

. Реалізація програм засовами мови JAVASCRIPT

**ХІД ВИКОНАННЯ РОБОТИ**

1. У звітному HTML-документі розмістити тему та постановку задачі лабораторної роботи №7.
2. Розмістити результати виконання та програмний код 2 пункту у звітному HTML-документі, записавши назви файлів HTML-документів.
3. Використовуючи сценарій на JavaScript, виконати завдання. яке приведене у Таблиці №1.
4. Кожному студенту виконати своє завдання.   
   Студенти, які мають непарні номери за списком групи виконують №варіантів : 1 ,3, 5, 7, 8, 9,10. Підключити загальний файл style.css для всіх варіантів
5. Студенти, які мають парні номери за списком групи виконують №варіантів : 2 ,4, 6, 7, 8, 9,10.
6. Підключити загальний файл **style.css** для всіх варіантів

Таблиці №1

|  |  |  |
| --- | --- | --- |
| №  варіанта | Завдання | Номери за списком групи |
| 1 | У звітному HTML-документі створити html-розміту, яка складається з наступних елементів : текст, кнопка, поле введення .    Натискання на кнопку "SHOW ME" має виводити значення з поля введення у консолі.  У звітному HTML-документі відобразити скрін програмного коду | Всі непарні номери за списком групи |
| 2 | У звітному HTML-документі створити html-розміту, яка складається з наступних елементів : текст, кнопка, два інпута (поля введення ).      Після натискання кнопки "SWAP ME" здійснюється обмін вмістом між двома інпутами.  Ви можете натиснути на неї кілька разів або вручну змінити вміст інпутів.  У звітному HTML-документі відобразити скрін програмного коду. | Всі парні номери за списком групи |
| 3 | У звітному HTML-документі створити html-розміту, яка складається з наступних елементів : текст, кнопка, інпут (поле введення ) .      Кнопка "**Приховати**" ховає текст , виводячи зірочки замість введеної інформації, замінює назву кнопки на "**Розкрити**", при повторному натисканні текст знову стає доступним і кнопка набуває початкового вигляду.  У звітному HTML-документі відобразити скрін програмного коду | Всі непарні номери за списком групи |
| 4 | У звітному HTML-документі створити html-розміту, яка складається з наступних елементів : текст, блок (div), дві кнопки    Результат:    Кнопка "**Зменшити**" робить квадрат менше на 15 пікселів.  Повторний натиск на кнопці **Зменшити**"   робить квадрат ще менше на 15 пікселів.    "**Збільшити**" – робить квадрат більше на 15 пікселів. Повторний натиск на кнопці "**Зменшити**"   робить квадрат ще більше на 15 пікселів.  У звітному HTML-документі відобразити скрін програмного коду |  |
| 5 | У звітному HTML-документі створити html-розміту, яка складається з наступних елементів :  - текст, з використанням селектора класу (class="taskTitle" ) ,  - div, з використанням селектора ідентифікатора (id="place”)    Додайте слухач кліку на window і визначте чи клікнув користувач у div з id="place".  Примітка: Якщо користувач клікнув на зеленому прямокутнику – у консолі виведе true. У протилежному випадку - false  У звітному HTML-документі відобразити скрін програмного коду | Всі непарні номери за списком групи |
| 6 | У звітному HTML-документі створити html-розміту, яка складається з наступних елементів : текст, маркирований список.    **Результат:**    Натиснувши кнопку "Подвоїти", збільшити значення  у кожному елементі списку у 2 рази. Повторний натиск на кнопці "Подвоїти", також збільшить значення у кожному елементі списку у 2 рази  У звітному HTML-документі відобразити скрін програмного коду | Всі парні номери за списком групи |
| 7 | TML містить список категорій ul#categories.  <ul id="categories">  <li class="item">  <h2>Animals</h2>  <ul>  <li>Cat</li>  <li>Hamster</li>  <li>Horse</li>  <li>Parrot</li>  </ul>  </li>  <li class="item">  <h2>Products</h2>  <ul>  <li>Bread</li>  <li>Prasley</li>  <li>Cheese</li>  </ul>  </li>  <li class="item">  <h2>Technologies</h2>  <ul>  <li>HTML</li>  <li>CSS</li>  <li>JavaScript</li>  <li>React</li>  <li>Node.js</li>  </ul>  </li> </ul>  Напиши скрипт, який:   1. Порахує і виведе в консоль кількість категорій в ul#categories, тобто елементів li.item. 2. Для кожного элемента li.item у списку ul#categories, знайде і виведе в консоль текст заголовку елемента (тегу <h2>) і кількість елементів в категорії (усіх <li>, вкладених в нього).   Для виконання цього завдання потрібно використати метод forEach() і властивості навігації по DOM.  В результаті, в консолі будуть виведені наступні повідомлення. | Всі непарні та парні номери за списком групи |
| 8 | Напиши скрипт управління формою логіна.  <form class="login-form">  <label>  Email  <input type="email" name="email" />  </label>  <label>  Password  <input type="password" name="password" />  </label>  <button type="submit">Log in</button> </form>   1. Обробка відправлення форми form.login-form повинна відбуватися за подією submit. 2. Під час відправлення форми сторінка не повинна перезавантажуватися. 3. Якщо при сабміті у формі є незаповнені поля, виводь alert з попередженням про те, що 'All form fields must be filled in'. Не додавай на інпути атрибут required, валідація має відбуватися саме через JS. 4. Якщо користувач заповнив усі поля і відправив форму, збери значення полів в об'єкт з двома властивостями, де ключ — це ім'я інпутів, а значення — відповідні значення цих інпутів, очищені від пробілів по краях. Для доступу до елементів форми використовуй властивість elements. 5. При сабміті форми виведи об'єкт із введеними даними в консоль і очисти значення полів форми методом reset. | Всі непарні та парні номери за списком групи |
| 9 | Напиши скрипт, який змінює колір фону елемента <body> через інлайн-стиль по кліку на button.change-color і задає це значення кольору текстовим вмістом для span.color.  <div class="widget">  <p>Background color: <span class="color">-</span></p>  <button type="button" class="change-color">Change color</button> </div>  Для генерування випадкового кольору використовуй функцію getRandomHexColor().  function getRandomHexColor() {  return `#${Math.floor(Math.random() \* 16777215)  .toString(16)  .padStart(6, 0)}`; }  Зверни увагу, що функція getRandomHexColor() повертає колір у hex-форматі, в той час як колір фону на <body> буде у форматі rgb. Це нормально й не потребує якихось правок. | Всі непарні та парні номери за списком групи |
| 10 | Напиши скрипт створення й очищення колекції елементів з наступним функціоналом.  Є **input**, у який користувач вводить бажану кількість елементів. Після натискання на кнопку **Create** має рендеритися (додаватися в DOM) колекція з відповідною кількістю елементів і очищатися значення в інпуті. При повторному натисканні на кнопку **Create** поверх старої колекції має рендеритись нова. Після натискання на кнопку **Destroy** колекція елементів має очищатися.  <div id="controls">  <input type="number" min="1" max="100" step="1" />  <button type="button" data-create>Create</button>  <button type="button" data-destroy>Destroy</button> </div>  <div id="boxes"></div>  Після натискання користувачем на кнопку **Create** треба провалідувати значення в input, воно має бути в межах від 1 до 100 включно. Тільки якщо воно задоволяє умову, мають додаватися нові <**div**> елементи в DOM.  Для рендеру елементів на сторінці створи функцію **createBoxes**(**amount**), яка приймає один параметр — число, що зберігає кількість елементів для рендеру. Функція має створювати стільки <**div**> елементів, скільки вказано в параметрі amount і додавати їх у DOM дочірніми елементами для **div**#**boxes**.   1. Розміри першого <**div**> елемента мають бути 30px на 30px. 2. Кожен наступний елемент повинен бути ширшим і вищим від попереднього на 10px. 3. Усі елементи повинні мати випадковий колір фону. Використовуй готову функцію **getRandomHexColor()**для отримання випадкового кольору.   function getRandomHexColor() {  return `#${Math.floor(Math.random() \* 16777215)  .toString(16)  .padStart(6, 0)}`; }  Для очищення колекції після натискання на кнопку **Destroy** створи функцію **destroyBoxes**(), яка очищає вміст div#**boxes**, у такий спосіб видаляючи всі створені елементи. | Всі непарні та парні номери за списком групи |

1. Розмістити програмний код та результати виконання у звітному HTML-документі.
2. Оформлення звітного HTML-документа лабораторної роботи №7 оформити на свій розсуд