Критерії приймання

- Створено репозиторій `goit-js-hw-06`.
- Домашня робота містить два посилання: на вихідні файли і робочу сторінку на `GitHub Pages`.
- Завдання виконані у точній відповідності до ТЗ (забороняється змінювати вихідний HTML завдання).
- В консолі відсутні помилки і попередження під час відкриття живої сторінки завдання.
- Імена змінних і функцій - зрозумілі та описові.
- Код відформатований за допомогою `Prettier`.

Стартові файли

Завантажуй стартові файли з готовою розміткою та підключеними файлами скриптів для кожного завдання. Скопіюй їх собі у проект.

Завдання 1

HTML 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.
Number of categories: 3

Category: Animals
Elements: 4

Category: Products
Elements: 3

Category: Technologies
Elements: 5

Завдання 2

HTML містить порожній список `ul#ingredients`.

```
<ul id="ingredients"></ul>
```

JavaScript містить масив рядків.

```
const ingredients = [
  "Potatoes",
  "Mushrooms",
```

```
  "Garlic",
  "Tomatos",
  "Herbs",
  "Condiments",
];
```

Напиши скрипт, який для кожного елемента масиву `ingredients`:

1. Створить окремий елемент `<li>`. Обов'язково використовуй метод `document.createElement()`.
2. Додасть назву інгредієнта як його текстовий вміст.
3. Додасть елементу клас `item`.
4. Після чого, вставить усі `<li>` за одну операцію у список `ul#ingredients`.

Завдання 3

Напиши скрипт для створення галереї зображень на підставі масиву даних. HTML містить список `ul.gallery`.

```
<ul class="gallery"></ul>
```

Використовуй масив об'єктів `images` для створення елементів `<img>`, вкладених в `<li>`. Для створення розмітки використовуй шаблонні рядки і метод `insertAdjacentHTML()`.

- Усі елементи галереї повинні додаватися в DOM за одну операцію додавання.
- Додай мінімальне оформлення галереї флексбоксами або грідами через CSS класи.

```
const images = [
  {
    url:
"https://images.pexels.com/photos/140134/pexels-photo-140134.jpeg?dpr=2&h=750&w=1260",
    alt: "White and Black Long Fur Cat",
  },
  {
    url:
"https://images.pexels.com/photos/213399/pexels-photo-213399.jpeg?dpr=2&h=750&w=1260",
    alt: "Orange and White Koi Fish Near Yellow Koi Fish",
  },
```

```
 {
   url:
"https://images.pexels.com/photos/219943/pexels-photo-219943.jpeg?dpr=2&h=750&w=1260",
   alt: "Group of Horses Running",
 },
];
```

Завдання 4

Лічильник складається зі спану і кнопок, які по кліку повинні збільшувати і зменшувати його значення на одиницю.

```
<div id="counter">
 <button type="button" data-action="decrement">-1</button>
 <span id="value">0</span>
 <button type="button" data-action="increment">+1</button>
</div>
```

- Створи змінну `counterValue`, в якій буде зберігатися поточне значення лічильника та ініціалізуй її значенням `0`.
- Додай слухачів кліків до кнопок, всередині яких збільшуй або зменшуй значення лічильника.
- Оновлюй інтерфейс новим значенням змінної `counterValue`.

Завдання 5

Напиши скрипт, який під час набору тексту в інпуті `input#name-input` (подія `input`), підставляє його поточне значення в `span#name-output`. Якщо інпут порожній, у спані повинен відображатися рядок `"Anonymous"`.

```
<input type="text" id="name-input" placeholder="Please enter your name" />
<h1>Hello, <span id="name-output">Anonymous</span>!</h1>
```

Завдання 6

Напиши скрипт, який під час втрати фокусу на інпуті (подія `blur`), перевіряє його вміст щодо правильної кількості введених символів.

```
<input
 type="text"
 id="validation-input"
 data-length="6"
 placeholder="Please enter 6 symbols"
```

/>

- Яка кількість символів повинна бути в інпуті, зазначається в його атрибуті `data-length`.
- Якщо введена правильна кількість символів, то `border` інпуту стає зеленим, якщо неправильна кількість - червоним.

Для додавання стилів використовуй CSS-класи `valid` і `invalid`, які ми вже додали у вихідні файли завдання.

```
#validation-input {
 border: 3px solid #bdbdbd;
}

#validation-input.valid {
 border-color: #4caf50;
}

#validation-input.invalid {
 border-color: #f44336;
}
```

Завдання 7

Напиши скрипт, який реагує на зміну значення `input#font-size-control` (подія `input`) і змінює інлайн-стиль `span#text`, оновлюючи властивість `font-size`. В результаті, перетягуючи повзунок, буде змінюватися розмір тексту.

```
<input id="font-size-control" type="range" min="16" max="96" />
<br />
<span id="text">Abracadabra!</span>
```

Завдання 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">Login</button>
</form>
```

1. Обробка відправлення форми `form.login-form` повинна відбуватися відповідно до події `submit`.
2. Під час відправлення форми сторінка не повинна перезавантажуватися.
3. Якщо у формі є незаповнені поля, виводь `alert` з попередженням про те, що всі поля повинні бути заповнені.
4. Якщо користувач заповнив усі поля і відправив форму, збери значення полів в об'єкт, де ім'я поля буде ім'ям властивості, а значення поля - значенням властивості. Для доступу до елементів форми використовуй властивість `elements`.
5. Виведи об'єкт із введеними даними в консоль і очисти значення полів форми методом `reset`.

Завдання 9

Напиши скрипт, який змінює кольори фону елемента `<body>` через інлайн-стиль по кліку на `button.change-color` і виводить значення кольору в `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)}`;
}
```

Завдання 10 (виконувати не обов'язково)

Напиши скрипт створення і очищення колекції елементів. Користувач вводить кількість елементів в `input` і натискає кнопку `Створити`, після чого рендериться колекція. Натисненням на кнопку `Очистити`, колекція елементів очищається.

```
<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>
```

Створи функцію `createBoxes(amount)`, яка приймає один параметр - число. Функція створює стільки `<div>`, скільки вказано в `amount` і додає їх у `div#boxes`.

1. Розміри найпершого `<div>` - 30px на 30px.
2. Кожен елемент після першого повинен бути ширшим і вищим від попереднього на 10px.
3. Всі елементи повинні мати випадковий колір фону у форматі HEX. Використовуй готову функцію `getRandomHexColor` для отримання кольору.

```
function getRandomHexColor() {
 return `#${Math.floor(Math.random() * 16777215)
  .toString(16)
  .padStart(6, 0)}`;
}
```

Створи функцію `destroyBoxes()`, яка очищає вміст `div#boxes`, у такий спосіб видаляючи всі створені елементи.