Завдання 1 - перемикач кольорів

Виконуй це завдання у файлах `01-color-switcher.html` і `01-color-switcher.js`. Подивися демо-відео роботи перемикача.

HTML містить кнопки «Start» і «Stop».

```
<button type="button" data-start>Start</button>
<button type="button" data-stop>Stop</button>
```

Напиши скрипт, який після натискання кнопки «Start», раз на секунду змінює колір фону `<body>` на випадкове значення, використовуючи інлайн стиль. Натисканням на кнопку «Stop» зміна кольору фону повинна зупинятися.

УВАГА

Враховуй, що на кнопку «Start» можна натиснути нескінченну кількість разів. Зроби так, щоб доки зміна теми запущена, кнопка «Start» була неактивною (disabled).

Для генерування випадкового кольору використовуй функцію `getRandomHexColor`.

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

Завдання 2 - таймер зворотного відліку

Виконуй це завдання у файлах `02-timer.html` і `02-timer.js`. Напиши скрипт таймера, який здійснює зворотний відлік до певної дати. Такий таймер може використовуватися у блогах та інтернет-магазинах, сторінках реєстрації подій, під час технічного обслуговування тощо. Подивися демо-відео роботи таймера.

Елементи інтерфейсу

HTML містить готову розмітку таймера, поля вибору кінцевої дати і кнопку, по кліку на яку, таймер повинен запускатися. Додай мінімальне оформлення елементів інтерфейсу.

```
<input type="text" id="datetime-picker" />
<button type="button" data-start>Start</button>

<div class="timer">
  <div class="field">
    <span class="value" data-days>00</span>
    <span class="label">Days</span>
  </div>
  <div class="field">
    <span class="value" data-hours>00</span>
    <span class="label">Hours</span>
  </div>
```

```
  <div class="field">
    <span class="value" data-minutes>00</span>
    <span class="label">Minutes</span>
  </div>
  <div class="field">
    <span class="value" data-seconds>00</span>
    <span class="label">Seconds</span>
  </div>
</div>
```

Бібліотека `flatpickr`

Використовуй бібліотеку flatpickr для того, щоб дозволити користувачеві кросбраузерно вибрати кінцеву дату і час в одному елементі інтерфейсу. Для того щоб підключити CSS код бібліотеки в проект, необхідно додати ще один імпорт, крім того, що описаний в документації.

```
// Описаний в документації
import flatpickr from "flatpickr";
// Додатковий імпорт стилів
import "flatpickr/dist/flatpickr.min.css";
```

Бібліотека очікує, що її ініціалізують на елементі `input[type="text"]`, тому ми додали до HTML документу поле `input#datetime-picker`.

```
<input type="text" id="datetime-picker" />
```

Другим аргументом функції `flatpickr(selector, options)` можна передати необов'язковий об'єкт параметрів. Ми підготували для тебе об'єкт, який потрібен для виконання завдання. Розберися, за що відповідає кожна властивість в документації «Options», і використовуй його у своєму коді.

```
const options = {
  enableTime: true,
  time_24hr: true,
  defaultDate: new Date(),
  minuteIncrement: 1,
  onClose(selectedDates) {
    console.log(selectedDates[0]);
  },
};
```

Вибір дати

Метод `onClose()` з об'єкта параметрів викликається щоразу під час закриття елемента інтерфейсу, який створює `flatpickr`. Саме у ньому варто обробляти дату, обрану користувачем. Параметр `selectedDates` - це масив обраних дат, тому ми беремо перший елемент.

- Якщо користувач вибрав дату в минулому, покажи `window.alert()` з текстом `"Please choose a date in the future"`.
- Якщо користувач вибрав валідну дату (в майбутньому), кнопка «Start» стає активною.
- Кнопка «Start» повинна бути неактивною доти, доки користувач не вибрав дату в майбутньому.
- Натисканням на кнопку «Start» починається відлік часу до обраної дати з моменту натискання.

Відлік часу

Натисканням на кнопку «Start» скрипт повинен обчислювати раз на секунду, скільки часу залишилось до вказаної дати, і оновлювати інтерфейс таймера, показуючи чотири цифри: дні, години, хвилини і секунди у форматі `xx:xx:xx:xx`.

- Кількість днів може складатися з більше, ніж двох цифр.
- Таймер повинен зупинятися, коли дійшов до кінцевої дати, тобто `00:00:00:00`.

НЕ БУДЕМО УСКЛАДНЮВАТИ

Якщо таймер запущений, для того щоб вибрати нову дату і перезапустити його - необхідно перезавантажити сторінку.

Для підрахунку значень використовуй готову функцію `convertMs`, де `ms` - різниця між кінцевою і поточною датою в мілісекундах.

```
function convertMs(ms) {
  // Number of milliseconds per unit of time
  const second = 1000;
  const minute = second * 60;
  const hour = minute * 60;
  const day = hour * 24;

  // Remaining days
  const days = Math.floor(ms / day);
  // Remaining hours
  const hours = Math.floor((ms % day) / hour);
  // Remaining minutes
  const minutes = Math.floor(((ms % day) % hour) / minute);
  // Remaining seconds
  const seconds = Math.floor((((ms % day) % hour) % minute) / second);

  return { days, hours, minutes, seconds };
}

console.log(convertMs(2000)); // {days: 0, hours: 0, minutes: 0, seconds: 2}
console.log(convertMs(140000)); // {days: 0, hours: 0, minutes: 2, seconds: 20}
console.log(convertMs(24140000)); // {days: 0, hours: 6 minutes: 42, seconds: 20}
```

Форматування часу

Функція `convertMs()` повертає об'єкт з розрахованим часом, що залишився до кінцевої дати. Зверни увагу, що вона не форматує результат. Тобто, якщо залишилося 4 хвилини або будь-якої іншої складової часу, то функція поверне `4`, а не `04`. В інтерфейсі таймера необхідно додавати `0`, якщо в числі менше двох символів. Напиши функцію `addLeadingZero(value)`, яка використовує метод `padStart()` і перед рендерингом інтефрейсу форматує значення.

Бібліотека повідомлень

УВАГА

Наступний функціонал не обов'язковий для здавання завдання, але буде хорошою додатковою практикою.

Для відображення повідомлень користувачеві, замість `window.alert()`, використовуй бібліотеку notiflix.

Завдання 3 - генератор промісів

Виконуй це завдання у файлах `03-promises.html` і `03-promises.js`. Подивися демо-відео роботи генератора промісів.

HTML містить розмітку форми, в поля якої користувач буде вводити першу затримку в мілісекундах, крок збільшення затримки для кожного промісу після першого і кількість промісів, яку необхідно створити.

```
<form class="form">
  <label>
    First delay (ms)
    <input type="number" name="delay" required />
  </label>
  <label>
    Delay step (ms)
    <input type="number" name="step" required />
  </label>
  <label>
    Amount
    <input type="number" name="amount" required />
  </label>
  <button type="submit">Create promises</button>
</form>
```

Напиши скрипт, який на момент сабміту форми викликає функцію `createPromise(position, delay)` стільки разів, скільки ввели в поле `amount`. Під час кожного виклику передай їй номер промісу (`position`), що створюється, і затримку, враховуючи першу затримку (`delay`), введену користувачем, і крок (`step`).

```
function createPromise(position, delay) {
  const shouldResolve = Math.random() > 0.3;
  if (shouldResolve) {
    // Fulfill
```

```
  } else {
    // Reject
  }
}
```

Доповни код функції `createPromise` таким чином, щоб вона повертала **один проміс**, який виконується або відхиляється через `delay` часу. Значенням промісу повинен бути об'єкт, в якому будуть властивості `position` і `delay` зі значеннями однойменних параметрів. Використовуй початковий код функції для вибору того, що потрібно зробити з промісом - виконати або відхилити.

```
createPromise(2, 1500)
  .then(({ position, delay }) => {
    console.log(`✅ Fulfilled promise ${position} in ${delay}ms`);
  })
  .catch(({ position, delay }) => {
    console.log(`❌ Rejected promise ${position} in ${delay}ms`);
  });
```

Бібліотека повідомлень

УВАГА

Наступний функціонал не обов'язковий для здавання завдання, але буде хорошою додатковою практикою.

Для відображення повідомлень користувачеві, замість `console.log()`, використовуй бібліотеку notiflix.