Лабораторна робота №2
Тема: регулярні вирази, робота з датами та одновимірними масивами у JavaScript з використанням Chrome DevTools

Мета роботи: навчитися створювати, форматувати й порівнювати дати в JavaScript; опрацювати типові операції з одновимірними масивами: перебір, пошук, сортування, фільтрація, агрегація; навчитися аналізувати часові ряди та будувати прості утиліти/візуалізації в консолі; освоїти інструменти розробника Google Chrome: Console, Sources → Snippets, Breakpoints, console.table, Performance (за бажанням).

Зміст роботи:

Завдання на лабораторну роботу

Завдання 1. Ретельно вивчити теорію:

https://uk.javascript.info/date
https://uk.javascript.info/array
https://uk.javascript.info/regexp-introduction
https://uk.javascript.info/regexp-methods
https://w3schoolsua.github.io/js/js_regexp.html#gsc.tab=0

Примітка: ключові тези теорії:

- Date: `new Date()`, `new Date(ms)`, `new Date(iso)`, методи: `getFullYear()`, `getMonth()` (0–11), `getDate()`, `getDay()` (0–6), `getTime()` (мс від 1970-01-01), `set*`-методи.
- Часові пояси та форматування: `Intl.DateTimeFormat(locale, options)`, параметри `timeZone`, `weekday`, `year`, `month`, `day`, `hour`, `minute`, `second`. Важливо: Date зберігає мить часу у UTC, а показує — у вашій TZ, якщо не вказано інше.
- Операції з датами: різниця в днях = `(d2 - d1) / (1000*60*60*24)`, нормалізація до півночі, перевірка вихідного (`getDay()===0||6`).
- Масиви: `map`, `filter`, `reduce`, `some`, `every`, `find`, `sort`, поширене копіювання `[...]`, порівняння/сортування за ключем.
- Консоль: `console.log`, `console.warn`, `console.error`, `console.time/timeEnd`, `console.table` (ідеально для масивів об'єктів).
- Регулярні вирази (RegExp)**:** створення /pattern/flags або new RegExp(). Основні елементи: ^ (початок), $ (кінець), . (будь-який символ), *, +, ?, групи (), квантифікатори {m,n}, клас символів [a-z], \d, \w, негативні класи [^...].
- DevTools: вкладка *Sources* → *Snippets* (зберігайте код задач як сніпети), *Breakpoints* для відлагодження.
- Пам'ятайте: місяці починаються з 0, `getMonth()` → 0…11.
- Порівнюйте дати за числом мілісекунд: `a - b`.
- Для різниці в днях спершу нормалізуйте час (`setHours(0,0,0,0)`).
- `sort` для дат: `arr.sort((a,b)=>a-b)`.
- Не плутайте локальний час і UTC — для форматування часових поясів завжди вказуйте `timeZone`.

УВАГА! ДЛЯ ВИКОНАННЯ ДАНОЇ ЛАБОРАТОРНОЇ РОБОТИ ВІДКРИЙТЕ **Chrome → DevTools (F12) → Sources → Snippets.** ДЛЯ КОЖНОГО ЗАВДАННЯ СТВОРІТЬ ОКРЕМИЙ СНІПЕТ, ЗАПУСТІТЬ ЙОГО КНОПКОЮ ▶️**,** РЕЗУЛЬТАТИ ДИВІТЬСЯ У **Console.**

Завдання 2. Напишіть функцію «Скільки днів до Нового року?». Опис: функція обчислює, скільки повних днів залишилось до 1 січня наступного року. Виведіть у консоль як число днів, так і красиво відформатовану цільову дату з `Intl`.

Методичні рекомендації:

- Створіть сьогоднішню дату `now` та дату `NY` = 1 січня `now.getFullYear()+1`.
- Обнуліть час (`setHours(0,0,0,0)`) для коректної різниці в днях.
- Форматуйте `NY` з локаллю `uk-UA`.

Приклад коду:

```
(function daysToNewYear(){
    const now = new Date();
    const NY = new Date(now.getFullYear() + 1, 0, 1);
    const msPerDay = 1000 * 60 * 60 * 24;
    const diffDays = Math.ceil((NY.setHours(0,0,0,0) - now.setHours(0,0,0,0)) / msPerDay);
    const fmt = new Intl.DateTimeFormat('uk-UA', { dateStyle: 'full' });
    console.log(`🎉 До ${fmt.format(NY)} залишилось: ${diffDays} дн.`);
})();
```

Завдання 3: Створіть функцію «Трекер звичок: підрахунок серій (streaks)». Задайте масив дат (у форматі ISO) коли студент робив вправи. Порахуйте найдовшу безперервну серію щоденних виконань.

Методичні рекомендації:

- Перетворіть рядки в `Date` і відсортуйте.
- Серію рахуйте, коли кожна наступна дата рівно на 1 день пізніша за попередню.
- Виведіть усі серії у `console.table` та найдовшу — окремо.

Приклад коду:

```
(function habitStreak(){
    const isoDates = [
    '2025-09-10','2025-09-11','2025-09-13','2025-09-14','2025-09-15','2025-09-17'
    ];
    const dates = isoDates.map(d=>new Date(d)).sort((a,b)=>a-b);
    const msDay = 86400000;
    let streaks = [], cur=[dates[0]];
    for(let i=1;i<dates.length;i++){
    const prev = dates[i-1], curD = dates[i];
    if ((curD.setHours(0,0,0,0) - prev.setHours(0,0,0,0)) === msDay) cur.push(curD);
    else { streaks.push(cur); cur=[curD]; }
    }
    if(cur.length) streaks.push(cur);
    const table = streaks.map((s,i)=>({#:`S${i+1}`, start:s[0].toISOString().slice(0,10), end:s.at(-
1).toISOString().slice(0,10), len:s.length}));
    console.table(table);
    const max = streaks.reduce((m,s)=> s.length>m.length?s:m, []);
    console.log(`🏆 Найдовша серія: ${max.length} днів (${max[0].toISOString().slice(0,10)} →
${max.at(-1).toISOString().slice(0,10)})`);
})();
```

Завдання 4. Створіть функцію «Генератор робочого календаря місяця». Згенеруйте масив усіх дат обраного місяця та відфільтруйте тільки робочі дні (пн–пт). Порахуйте їх кількість та виведіть календар у вигляді таблиці.

Методичні рекомендації:

- Вхід: рік і місяць (0–11). Використовуйте цикл, поки місяць збігається.
- Робочі: `getDay()` у [1..5]. Виводьте `weekday`, `date`.

- Для візуалізації — `console.table`.

Приклад коду:

```
(function workdaysOfMonth(year=2025, month=8){ // 0=січень; 8=вересень
    const res=[]; let d=new Date(year, month, 1);
    const fmt = new Intl.DateTimeFormat('uk-UA', { weekday:'long', year:'numeric', month:'long', day:'2-digit'
});
    while(d.getMonth()===month){
    const weekday = d.getDay();
    res.push({ date: fmt.format(d), weekday, isWorkday: weekday>=1 && weekday<=5 });
    d = new Date(d.getFullYear(), d.getMonth(), d.getDate()+1);
    }
    const work = res.filter(x=>x.isWorkday);
    console.table(work.map(({date,weekday})=>({date, weekday})));
    console.log(`🧮 Робочих днів: ${work.length}`);
})();
```

Завдання 5. Створіть функцію «Подорож у часі: конвертація часових поясів». Для списку дат (ISO) виведіть одночасно локальний час і час у Нью-Йорку та Токіо. Порівняйте різницю в годинах.

Методичні рекомендації:

- Використовуйте `Intl.DateTimeFormat` з `timeZone: 'America/New_York'` та `'Asia/Tokyo'`.
- Створіть зручну функцію `fmt(date, tz)`.

Приклад коду:

```
(function timezones(){
    const iso = ['2025-03-30T09:00:00Z','2025-07-01T09:00:00Z'];
    const f = (tz)=> new Intl.DateTimeFormat('uk-UA', { dateStyle:'medium', timeStyle:'short', timeZone: tz
});
    const local = f(Intl.DateTimeFormat().resolvedOptions().timeZone);
    const ny = f('America/New_York');
    const tokyo = f('Asia/Tokyo');
    const rows = iso.map(s=>{
    const d = new Date(s);
    return {
    iso: s,
    local: local.format(d),
    ny: ny.format(d),
    tokyo: tokyo.format(d)
    };
    });
    console.table(rows);
})();
```

Завдання 6. Створіть функцію «П’ятниці 13-го: пошук у діапазоні років». Знайдіть усі дати «п’ятниця 13-го» між роками Y1 і Y2 (включно). Виведіть список у таблиці та їхню кількість.

Методичні рекомендації:

- Ітеруйтесь по роках і місяцях, створюйте `new Date(y, m, 13)`.
- Перевіряйте `getDay()===5` (п’ятниця).

Приклад коду:

```
(function friday13(Y1=2020, Y2=2030){
    const res=[];
    for(let y=Y1;y<=Y2;y++){
    for(let m=0;m<12;m++){
    const d = new Date(y,m,13);
    if(d.getDay()===5){
    res.push({year:y, month:m+1, date: d.toISOString().slice(0,10)});
    }
    }
```

```
    }
    console.table(res);
console.log(`🔎 Знайдено: ${res.length} п’ятниць 13-го між ${Y1} і ${Y2}.`);
```

Завдання 7. Створіть функцію «Валідатор подій: перевірка правильності дат і назв за допомогою регулярних виразів». Функція повинна приймати масив рядків у форматі "YYYY-MM-DD: Назва події". Потрібно виділити дату та назву через RegExp, перевірити коректність дати та вивести у консоль таблицю з полями date, title, isValidDate.

Методичні рекомендації:

- Шаблон із групами: `^(\\d{4})-(\\d{2})-(\\d{2}):\\s*(.+)$`.
- Перевіряйте дату через `new Date(year, month-1, day)` і звіряйте `getFullYear()`, `getMonth()`, `getDate()`.
- Результат виведіть `console.table`.

Приклад коду:

```
(function regexEventValidator(){
    const events = [
    "2025-09-21: Лекція з JavaScript",
    "2025-09-31: Консультація",
    "2025-10-05: Екзамен"
    ];
    const pattern = /^(\d{4})-(\d{2})-(\d{2}):\s*(.+)$/;
    const rows = events.map(str=>{
    const m = str.match(pattern);
    if(!m) return { raw:str, date:null, title:null, isValidDate:false };
    const [_, y, mo, d, title] = m;
    const year = +y, month = +mo, day = +d;
    const dateObj = new Date(year, month-1, day);
    const isValid = dateObj.getFullYear()===year && dateObj.getMonth()===month-1 &&
dateObj.getDate()===day;
    return { date:`${year}-${mo}-${d}`, title, isValidDate:isValid };
    });
    console.table(rows);
})();
```

Завдання 8. Закомітити. Зайти на https://git.ztu.edu.ua, вказавши при реєстрації логін від локальної мережі Державного університету «Житомирська політехніка». Закомітити виконані завдання до власного репозиторію Frontend в папку Lab2.

Завдання 9. Надати доступ maintainer до створеного репозиторію викладачам:

@pzs_gms Граф Марина Сергіївна
@kkn_zhvv Желізко Віктор Вікторович
@kkn_oai Оринчак Андрій Іванович
@kkn_fdv Фуріхата Денис Вікторович

Додаткові завдання для закріплення пройденого матеріалу:

Завдання Д1: Створіть функцію «Консольний Pomodoro-таймер». Реалізуйте простий Pomodoro (25 хв фокус, 5 хв перерва) у консолі: зворотний відлік, зміна статусу, звуковий сигнал через `alert()` або `console.warn()`.

Методичні рекомендації:

- Використовуйте `setInterval` (для демонстрації можете масштабувати хвилину до 5–10 секунд).
- Форматуйте `мм:сс`.
- Зупинка через `clearInterval`.

Приклад коду (скорочений цикл: 10с роботи / 5с перерва):

```
(function pomodoro(){
    let phase='focus', seconds=10; // змініть на 25*60 та 5*60 для реального режиму
```

```
    const tick = setInterval(()=>{
    seconds--;
    const mm = String(Math.floor(seconds/60)).padStart(2,'0');
    const ss = String(seconds%60).padStart(2,'0');
    console.log(`${phase.toUpperCase()} ${mm}:${ss}`);
    if(seconds<=0){
    if(phase==='focus'){ phase='break'; seconds=5; console.warn('🔔 Перерва!'); }
    else { clearInterval(tick); console.warn('☑ Сеанс завершено'); }
    }
    }, 1000);
})();
```

Завдання Д2. Створіть функцію «Аналіз логів: середній інтервал та найактивніша година». Створіть початковий масив дат (масив міток часу ISO; наприклад, кліки/запуски). Обчисліть середній інтервал між подіями та визначте годину доби з найбільшою кількістю подій.

Методичні рекомендації:

- Відсортуйте події, різниці сумуйте та поділіть на `n-1`.
- Побудуйте гістограму по `getHours()` у вигляді об’єкта `{hour:count}`.
- Виведіть табличку та інсайт у консоль.

Приклад коду:

```
(function logAnalysis(){
    const iso = ['2025-09-21T08:10:00Z','2025-09-21T08:12:30Z','2025-09-21T10:00:00Z','2025-09-
21T10:10:00Z','2025-09-21T10:11:00Z'];
    const dates = iso.map(s=>new Date(s)).sort((a,b)=>a-b);
    let sum=0; for(let i=1;i<dates.length;i++) sum += dates[i]-dates[i-1];
    const avgMs = dates.length>1 ? sum/(dates.length-1) : 0;
    const histogram = Array.from({length:24},(_,h)=>({hour:h,count:0}));
    dates.forEach(d=> histogram[d.getHours()].count++ );
    console.table(histogram);
    console.log(`⏱ Середній інтервал: ${(avgMs/1000).toFixed(1)} c`);
    const top = histogram.reduce((m,x)=> x.count>m.count?x:m, {count:-1});
    console.log(`🕒 Найактивніша година: ${top.hour}:00 (подій: ${top.count})`);
})();
```

Завдання Д3. Створіть функцію «Менеджер подій: сортування та групування по тижнях». Є масив подій {title, dateISO}. Відсортуйте за датою зростання, згрупуйте за **номером тижня року**, виведіть результат console.table.

Методичні рекомендації:

- Номер тижня обчисліть правило ISO-8601 (тиждень починається в понеділок). Для спрощення можна використати наближення.
- В таблиці покажіть `week`, `date`, `title`.

Приклад коду (спрощений номер тижня):

```
(function eventsManager(){
    const events = [
    {title:'Дедлайн ЛР', dateISO:'2025-09-22'},
    {title:'Презентація', dateISO:'2025-09-25'},
    {title:'Консультація', dateISO:'2025-09-29'}
    ];
    const toWeek = d=>{ // спрощено: ISO тиждень ≈ (день року / 7)
    const yearStart = new Date(d.getFullYear(),0,1);
    const dayOfYear = Math.floor((d - yearStart)/(86400000)) + 1;
    return Math.ceil(dayOfYear/7);
    };
    const rows = events
    .map(e=>({ ...e, date:new Date(e.dateISO) }))
    .sort((a,b)=>a.date-b.date)
```

```
    .map(e=>({ week: toWeek(e.date), date: e.date.toISOString().slice(0,10), title:e.title }));
    console.table(rows);
})();
```

ДОМАШНЄ ЗАВДАННЯ, ОЦІНЮЄТЬСЯ ОКРЕМИМИ ДОДАТКОВИМИ БАЛАМИ

Завдання: Зібрати всі реалізовані сніпети в єдиний невеликий веб-проєкт.

Опис:

1. **Структура файлів**:
   - index.html — головна сторінка.
   - style.css — власні стилі.
   - scripts/ — папка з окремими JS-файлами для кожного завдання (наприклад, task1.js, task2.js ...).
2. **HTML:**
   - Використати Bootstrap (рекомендовано підключити через CDN, наприклад Bootstrap 5).
   - Створити зручну навігацію (navbar) з пунктами «Завдання 1», «Завдання 2» … «Завдання 9», «Міні-завдання».
   - Додати секції-карточки ( ) для кожного завдання з описом та кнопкою «Запустити».
3. **CSS:**
   - Власним стилем задати кольорову гаму (наприклад, світлий фон, яскраві акценти на кнопках).
   - Використати :hover для інтерактивності.
   - Додати відступи, тіні та закруглені кути карток.
4. **JS:**
   - Для кожного завдання додати кнопку, яка при натисканні запускає відповідний код (імпортований із scripts/taskN.js).
   - Результати виводити у консоль (як і у лабораторних завданнях).
   - Додати можливість показати результат у <pre> або на сторінці (опційно).

Вимоги до оформлення:

- Кожна картка має заголовок (назва завдання), короткий опис і кнопку «Запустити».
- На головній сторінці — привітальний блок (hero) з назвою «Лабораторна робота: Дати, Масиви та RegExp, Шевченко Тарас, група КН-32-1» (вказати власне прізвище та ім’я та вірну групу).
- Використати компоненти Bootstrap: navbar, card, button, container, row, col.

Результат:

У вас має вийти невеликий «навчальний сайт», де кожне завдання оформлене як окрема картка з можливістю запуску коду. Це буде візуально привабливий і структурований спосіб представлення всіх ваших сніпетів.

Методичні рекомендації до виконання домашнього завдання: зібрати сніпети у міні-дашборд (HTML + CSS + JS).

Мета домашнього завдання: перетворити консольні сніпети на о веб-інтерфейс із кнопками запуску, підсвіткою результатів та акуратним макетом на Bootstrap 5**.**

Вимоги до результату

- Окремі файли: index.html, styles.css, app.js (та за бажанням modules/*.js).
- Дизайн: використати Bootstrap 5 (CDN) + одну тему Bootswatch (рекомендую *Cosmo* або *Minty*) та Bootstrap Icons.
- Структура UI:
  - Верхня Navbar з назвою «Time & Data Lab» і перемикачем темного режиму (клас data-bs-theme="dark").
  - Accordion: секції для кожного завдання (1–9) + блок «RegExp міні».
  - У кожній секції: короткий опис, кнопка Run, панель Output (преформатований <pre>), і (для табличних задач) контейнер для HTML-таблиці.
  - Toast або Alert для повідомлень про помилки.
- Функціонал: натиснув кнопку — викликається відповідна функція зі сніпета; результати виводяться у <pre> і (де доречно) у таблицю. Паралельно залишайте лог у Console.

Структура проєкту

project/
├─ index.html
├─ styles.css
├─ app.js
└─ modules/
├─ dates.js // сюди винести функції задач 1,3,7,8
├─ arrays.js // задачі 2,5,6
└─ regex.js // завдання 9 + міні-задачі

Підключення бібліотек (CDN)

- Bootstrap 5:
  - CSS: https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css
  - JS: https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js
- Bootswatch тема (опціонально): https://cdn.jsdelivr.net/npm/bootswatch@5.3.3/dist/cosmo/bootstrap.min.css
- Bootstrap Icons: https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css

Порада: підключайте тільки одну тему — або звичайний Bootstrap, або Bootswatch (щоб стилі не конфліктували).

Шаблон index.html, базові стилі styles.css, логіка app.js (зв'язування кнопок з функціями), а також приклад винесення сніпетів у модулі modules/dates.js для Завдання 1 можна завантажити з папки projectLab2DZ **(**За аналогією до файлу dates.js перенесіть решту сніпетів у відповідні модулі та експортуйте по одній функції на завдання. Функція має повертати дані (об'єкт/рядок), які ви виводите у <pre>).

Як об'єднати табличні результати

Для задач із console.table (2,3,5,6,7,8,9) можна зробити допоміжну функцію рендеру в HTML-таблицю:

```
function renderTable(rows, mount){
    if(!rows || !rows.length){ mount.innerHTML = '<em>Немає даних</em>'; return; }
    const cols = Object.keys(rows[0]);
    const thead = `<thead><tr>${cols.map(c=>`<th>${c}</th>`).join('')}</tr></thead>`;
    const tbody =
`<tbody>${rows.map(r=>`<tr>${cols.map(c=>`<td>${r[c]}</td>`).join('')}</tr>`).join('')}</tbody>`;
    mount.innerHTML = `<table class="table table-sm table-striped table-
hover">${thead}${tbody}</table>`;
}
```