Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Практична робота № 5

з курсу: «*Основи Веб-програмування*»»

**Виконала:**  
студентка 2-го курсу,  
групи ТВ-32  
Чайка Олеся Ігорівна

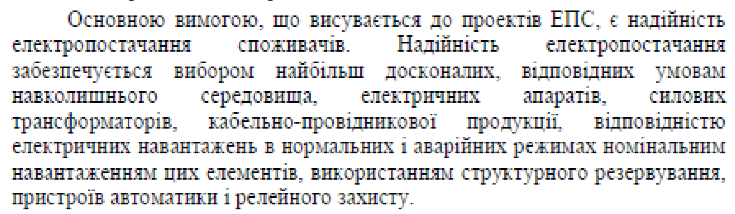
Посилання на GitHub репозиторій: https://github.com/LesiaChaika150/web-programming-practice

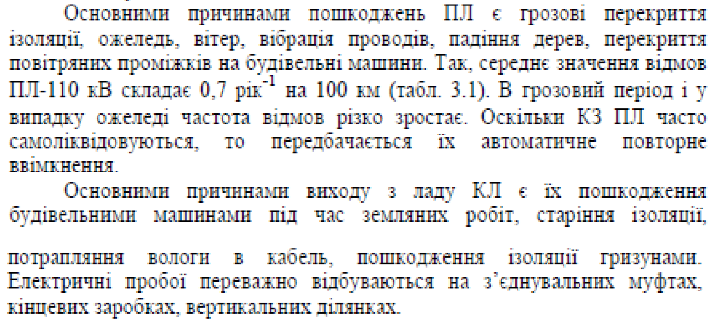
**Перевірив:**

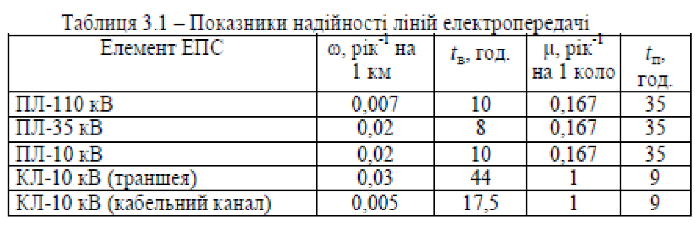
Недашківський О.Л.

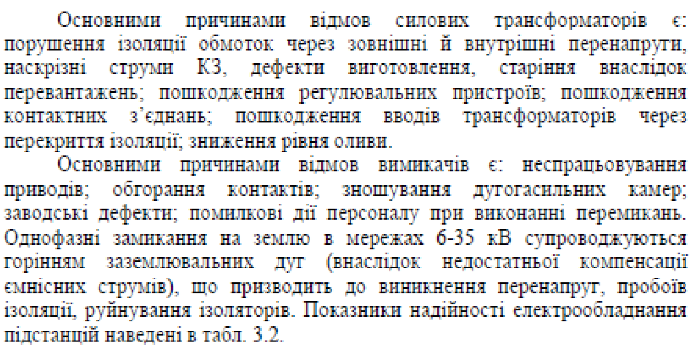
Київ 2024/2025

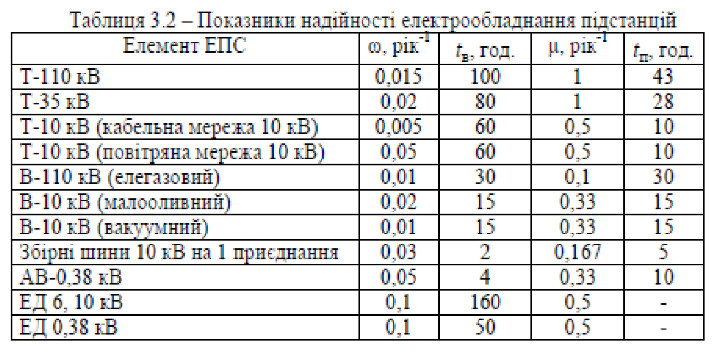
Практична робота № 5

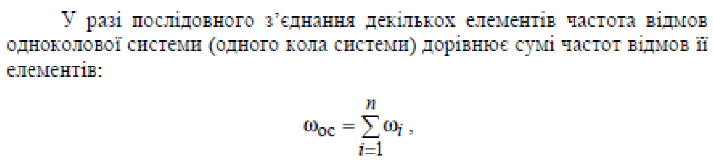




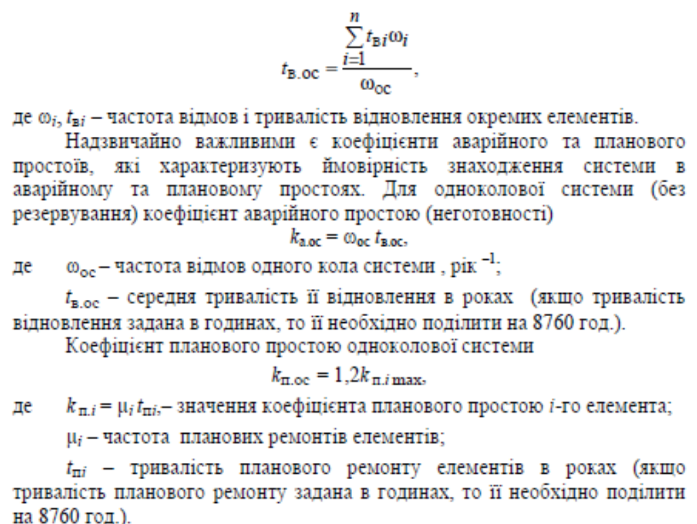


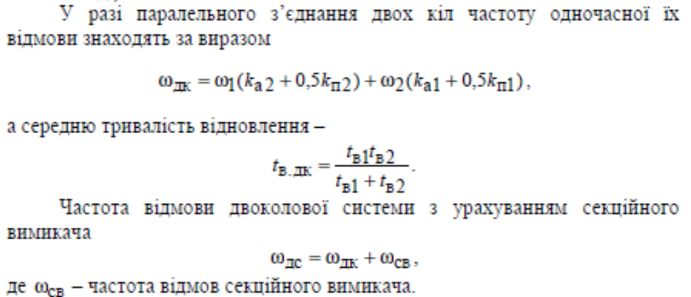












**Завдання:**   
 Створіть веб калькулятор для порівняння надійності одноколової та двоколової систем  електропередачі та розрахунку збитків від перерв електропостачання у разі застосування  одно трансформаторної ГТП у складі:

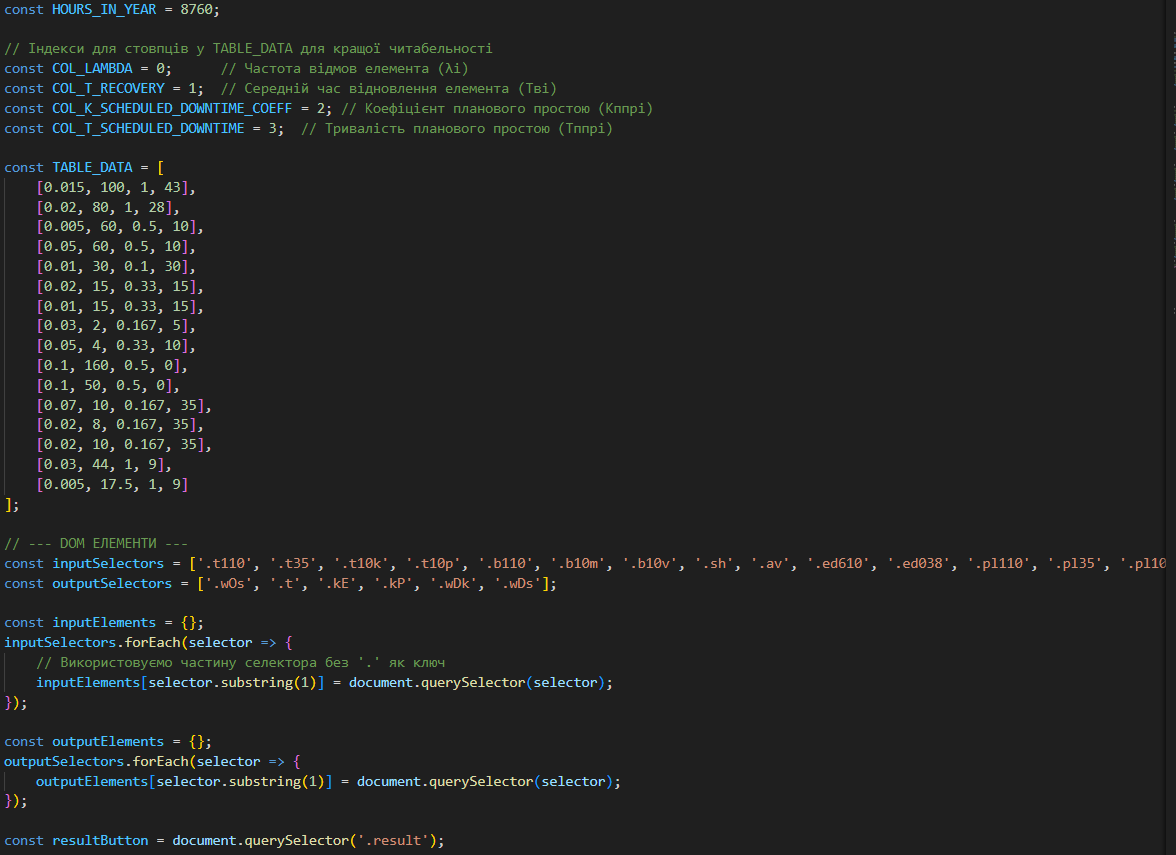
1. Порівняти надійність одноколової та двоколової систем електропередачі;

2. Розрахувати збитки від перерв електропостачання у разі застосування  одно трансформаторної ГПП.

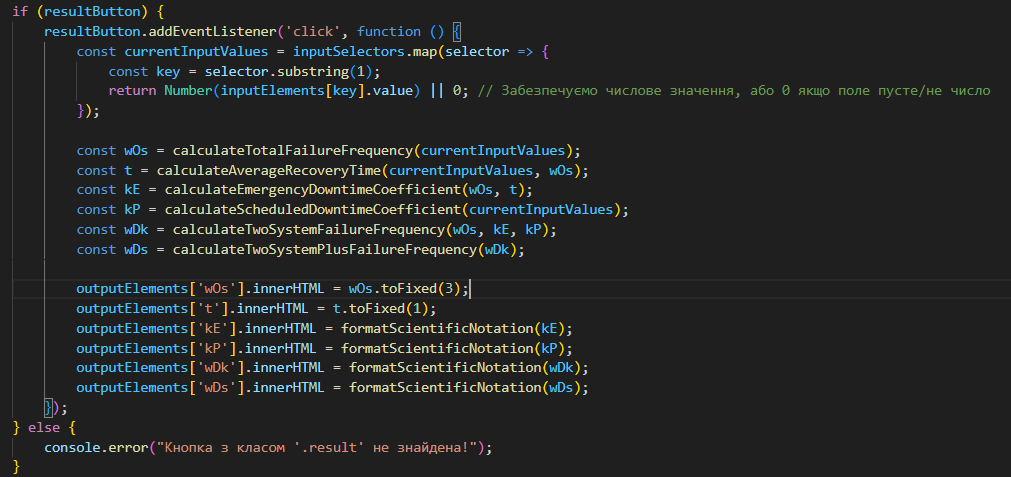
**Частина 1**

**Хід виконання:**

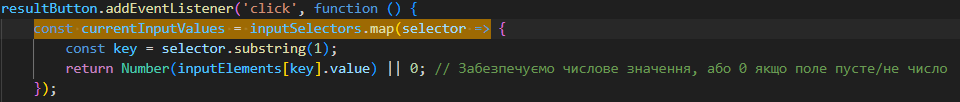
1. **Ініціалізація та підготовка (виконується при завантаженні скрипта)**

**Константи (HOURS\_IN\_YEAR, TABLE\_DATA, COL\_...):** На початку скрипта визначаються незмінні значення. TABLE\_DATA містить основні коефіцієнти для розрахунків. Константи COL\_... роблять код читабельнішим, дозволяючи звертатися до стовпців таблиці за осмисленими іменами замість числових індексів

**DOM Елементи:** Скрипт знаходить усі необхідні HTML-елементи (поля вводу, поля для виводу результатів, кнопку "Розрахувати") і зберігає посилання на них у об'єктах inputElements та outputElements. Це робиться один раз при завантаженні, щоб не шукати їх щоразу при кліку, що оптимізує продуктивність.

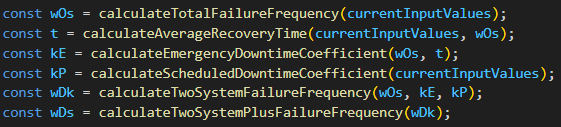
Це встановлює "слухача" на кнопку .result. Код всередині функції function () { ... } не виконується одразу. Він чекає, поки користувач не натисне на кнопку "Розрахувати".Перевірка if (resultButton) важлива, щоб уникнути помилки, якщо HTML-елемент кнопки з якихось причин не буде знайдено на сторінці.

1. **Головна логіка: що відбувається після кліку на кнопку "Розрахувати"**



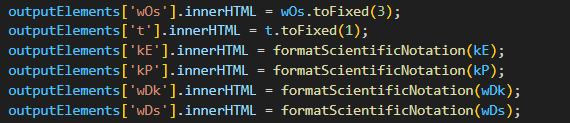
Створюється масив currentInputValues.Метод .map() проходить по кожному селектору з inputSelectors. Для кожного селектора береться відповідний DOM-елемент з inputElements. inputElements[key].value отримує текстове значення, введене користувачем у поле. Number(...) перетворює це текстове значення на число.|| 0 гарантує, що якщо поле порожнє або містить нечислове значення (що призведе до NaN після Number()), то замість NaN буде використано 0. Це важливо для коректності подальших розрахунків.В результаті currentInputValues стає масивом чисел, які ввів користувач (або нулів, якщо поля порожні). Порядок чисел у цьому масиві відповідає порядку елементів у TABLE\_DATA.

1. **Послідовний виклик функцій розрахунку**

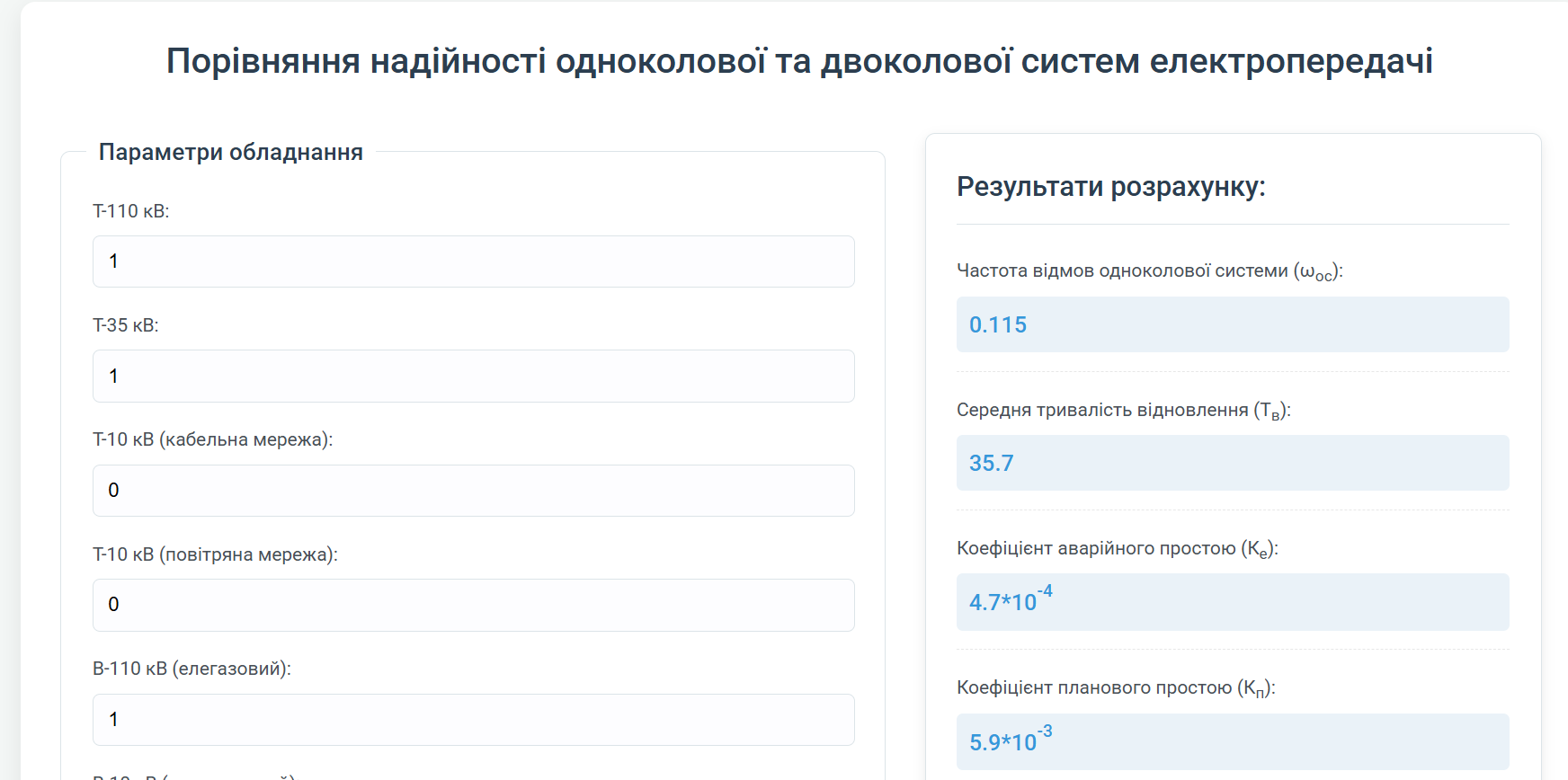
Тут відбувається власне магія розрахунків. Кожна функція виконує певну частину обчислень:

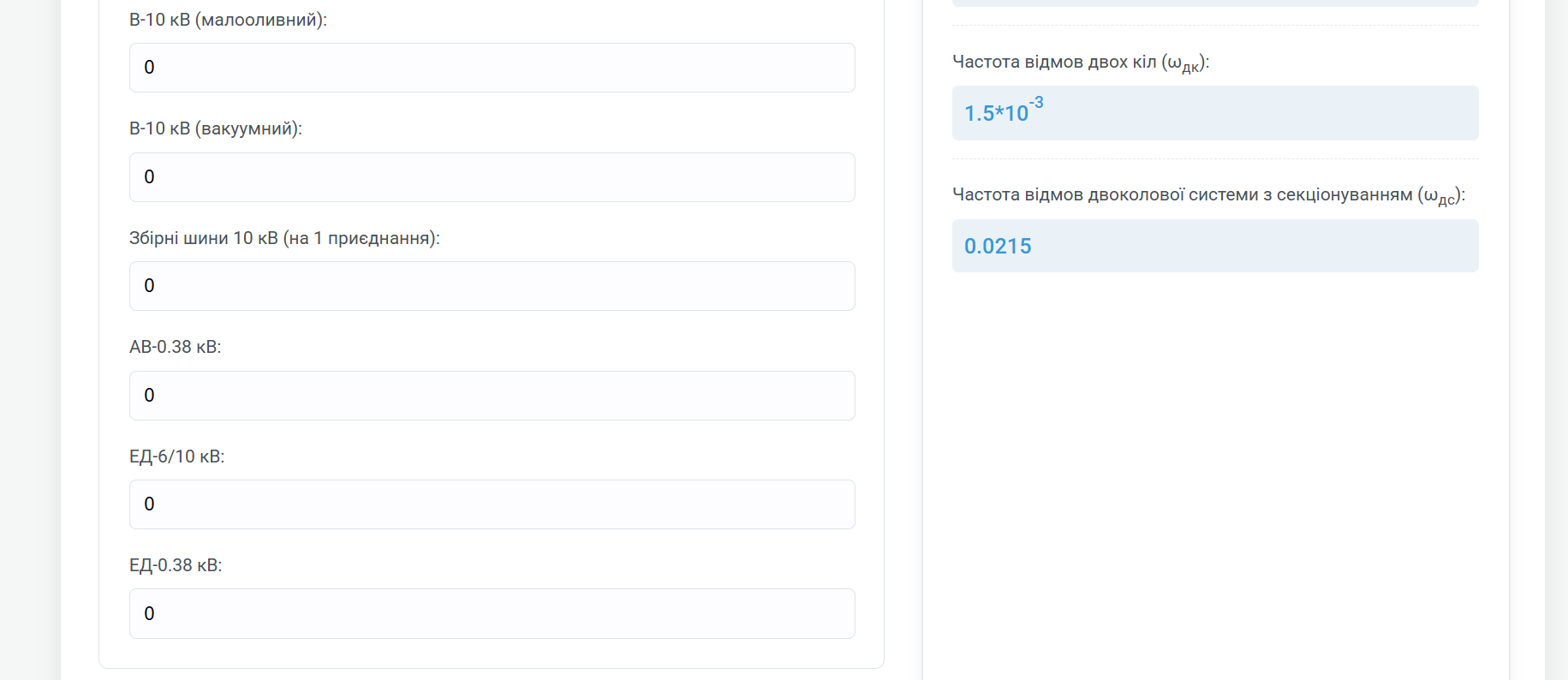
* calculateTotalFailureFrequency(currentInputValues): Розраховує сумарну частоту відмов системи (ω\_ос), використовуючи введені користувачем значення та дані з TABLE\_DATA.
* calculateAverageRecoveryTime(currentInputValues, wOs): Розраховує середній час відновлення (Т\_в). Для цього їй потрібні як введені значення, так і вже розрахована wOs.
* calculateEmergencyDowntimeCoefficient(wOs, t): Розраховує коефіцієнт аварійного простою (К\_е), використовуючи результати попередніх двох розрахунків.
* calculateScheduledDowntimeCoefficient(currentInputValues): Розраховує коефіцієнт планового простою (К\_п).
* calculateTwoSystemFailureFrequency(wOs, kE, kP): Розраховує частоту відмов для двосистемної конфігурації (ω\_дк).calculateTwoSystemPlusFailureFrequency(wDk): Розраховує фінальну частоту відмов (ω\_дс).

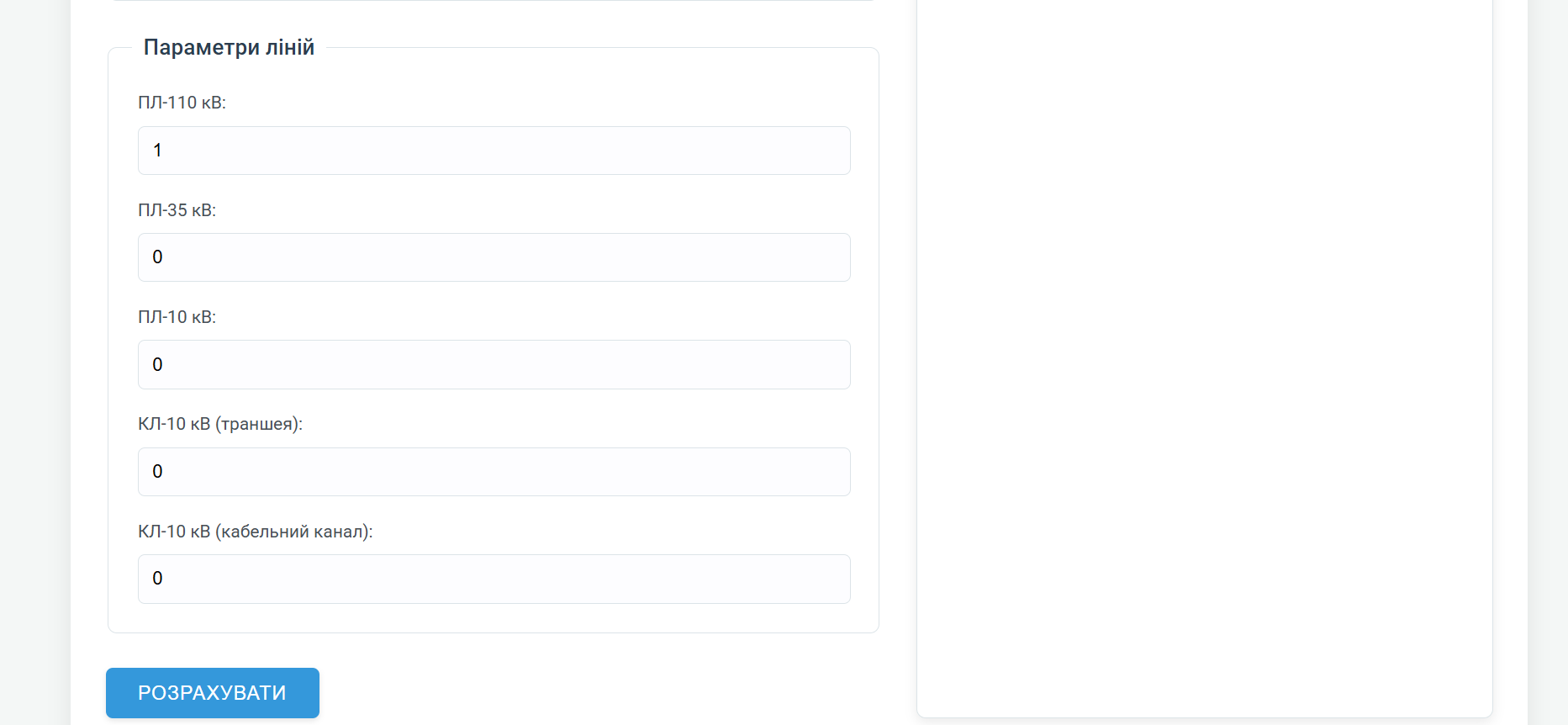
1. **Відображення результатів**

****

**Результат виконання:**







**Перевірка:**

**Вхідні дані для перевірки:**

* Кількість елемента типу 0: n\_0 = 1
* Кількість елемента типу 1: n\_1 = 1
* Кількість елемента типу 2: n\_2 = 0
* Кількість елемента типу 3: n\_3 = 0
* Кількість елемента типу 4: n\_4 = 1
* ... (решта до типу 10 - нулі)
* Кількість елемента типу 11: n\_11 = 1
* ... (решта нулі)

**Константи та табличні дані:**

* HOURS\_IN\_YEAR = 8760
* TABLE\_DATA (витягуємо дані за індексами, що відповідають ненульовим кількостям):

Для елемента типу 0: λ\_0 = 0.015, T\_в0 = 100, К\_ппрі0 = 1, Т\_ппрі0 = 43

Для елемента типу 1: λ\_1 = 0.02, T\_в1 = 80, К\_ппрі1 = 1, Т\_ппрі1 = 28

Для елемента типу 4: λ\_4 = 0.01, T\_в4 = 30, К\_ппрі4 = 0.1, Т\_ппрі4 = 30

Для елемента типу 11: λ\_11 = 0.07, T\_в11 = 10, К\_ппрі11 = 0.167, Т\_ппрі11 = 35

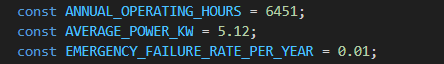
**Розрахунки:**

1. **Сумарна частота відмов системи:**  
   Формула: ω\_ос = Σ (n\_i \* λ\_i) для всіх i від 0 до 15.  
   ω\_ос = (n\_0 \* λ\_0) + (n\_1 \* λ\_1) + (n\_4 \* λ\_4) + (n\_11 \* λ\_11) (інші доданки дорівнюють нулю)  
   ω\_ос = (1 \* 0.015) + (1 \* 0.02) + (1 \* 0.01) + (1 \* 0.07)  
   ω\_ос = 0.015 + 0.020 + 0.010 + 0.070  
   ω\_ос = 0.115  
   *Очікуваний результат калькулятора (округлення до 3 знаків):* 0.115
2. **Середній час відновлення системи:**  
   Формула: Т\_в = ( Σ (n\_i \* λ\_i \* T\_вi) ) / ω\_ос  
   Чисельник: (n\_0 \* λ\_0 \* T\_в0) + (n\_1 \* λ\_1 \* T\_в1) + (n\_4 \* λ\_4 \* T\_в4) + (n\_11 \* λ\_11 \* T\_в11)  
   = (1 \* 0.015 \* 100) + (1 \* 0.02 \* 80) + (1 \* 0.01 \* 30) + (1 \* 0.07 \* 10)  
   = 1.5 + 1.6 + 0.3 + 0.7  
   = 4.1  
   Т\_в = 4.1 / 0.115 ≈ 35.652173913  
   *Очікуваний результат калькулятора (округлення до 1 знаку):* 35.7
3. **Коефіцієнт аварійного простою:**  
   Формула: К\_е = (ω\_ос \* Т\_в) / HOURS\_IN\_YEAR  
   Використовуємо чисельник з попереднього кроку для точності: (Σ (n\_i \* λ\_i \* T\_вi)) / HOURS\_IN\_YEAR  
   К\_е = 4.1 / 8760 ≈ 0.00046803653  
   *Очікуваний результат калькулятора (наукова нотація, 1 знак після коми для мантиси):* 4.7\*10⁻⁴
4. **Коефіцієнт планового простою:**  
   Формула: К\_п = (1.2 \* max\_i(К\_ппрі\_i \* Т\_ппрі\_i, де n\_i > 0)) / HOURS\_IN\_YEAR  
   Розраховуємо К\_ппрі\_i \* Т\_ппрі\_i для елементів, де n\_i > 0:
   * Елемент 0: 1 \* 43 = 43
   * Елемент 1: 1 \* 28 = 28
   * Елемент 4: 0.1 \* 30 = 3
   * Елемент 11: 0.167 \* 35 = 5.845  
     max(43, 28, 3, 5.845) = 43  
     К\_п = (1.2 \* 43) / 8760 = 51.6 / 8760 ≈ 0.00589041095  
     *Очікуваний результат калькулятора (наукова нотація, 1 знак після коми для мантиси):* 5.9\*10⁻³
5. **Частота відмов для двосистемної конфігурації:**  
   Формула: ω\_дк = 2 \* ω\_ос \* (К\_е + К\_п)  
   Використовуємо розраховані значення К\_е та К\_п:  
   К\_е + К\_п ≈ 0.00046803653 + 0.00589041095 ≈ 0.00635844748  
   ω\_дк = 2 \* 0.115 \* (0.00635844748)  
   ω\_дк = 0.23 \* 0.00635844748 ≈ 0.00146244292
6. **Частота відмов для двосистемної конфігурації з урахуванням додаткового фактора:**  
   Формула: ω\_дс = ω\_дк + 0.02  
   ω\_дс ≈ 0.00146244292 + 0.02 = 0.02146244292  
   *Очікуваний результат калькулятора (округлення до 4 знаків):* 0.0215

**Частина 2**

**Хід виконання:**

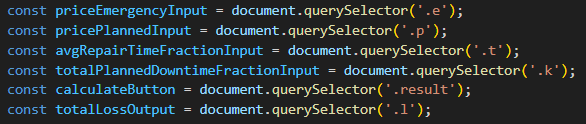
**Визначення глобальних констант:**



Ці рядки визначають три незмінні (константні) значення, які використовуються у формулах розрахунку збитків:

* ANNUAL\_OPERATING\_HOURS: Кількість годин, яку споживач працює протягом року.
* AVERAGE\_POWER\_KW: Середня потужність, яку споживає споживач, в кіловатах.
* EMERGENCY\_FAILURE\_RATE\_PER\_YEAR: Очікувана частота аварійних відключень на рік (наприклад, 0.01 означає 1 відключення на 100 років або 1% ймовірності відключення за рік, залежно від інтерпретації, але в даній формулі це частота, 1/рік).

**Пошук та збереження посилань на HTML-елементи:**



Скрипт знаходить на HTML-сторінці елементи за їхніми CSS-класами:

* .e: Поле для введення ціни аварійних вимкнень.
* .p: Поле для введення ціни планових вимкнень.
* .t: Поле для введення середнього часу відновлення аварії (як частки року).
* .k: Поле для введення сумарного часу планових відключень (як частки року).
* .result: Кнопка "Розрахувати".
* .l: Елемент <span> для відображення розрахованої суми збитків.  
  Посилання на ці елементи зберігаються у відповідних константах. Це робиться один раз при завантаженні скрипта, щоб потім швидко звертатися до цих елементів, не шукаючи їх щоразу.

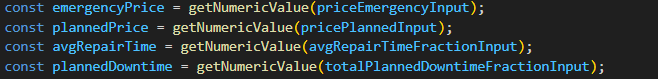
**Очікування дії користувача: налаштування обробника події для кнопки**



Цей блок коду перевіряє, чи була кнопка з класом .result знайдена (if (calculateButton)). Якщо так, то на неї "вішається" слухач події click.

Це означає, що скрипт буде чекати, поки користувач не натисне на кнопку "Розрахувати". Код, що знаходиться всередині анонімної функції function() { ... }, не виконується одразу, а лише після того, як відбудеться подія кліку.

**Головна логіка: виконання при натисканні кнопки "Розрахувати"**



Для кожного з чотирьох полів вводу викликається допоміжна функція getNumericValue().**Функція getNumericValue(inputElement):** (детальніше про неї нижче) Її завдання – взяти текстове значення з поля вводу (inputElement.value), перетворити його на число (Number()) і повернути. Якщо поле порожнє або містить нечислове значення, функція поверне 0.

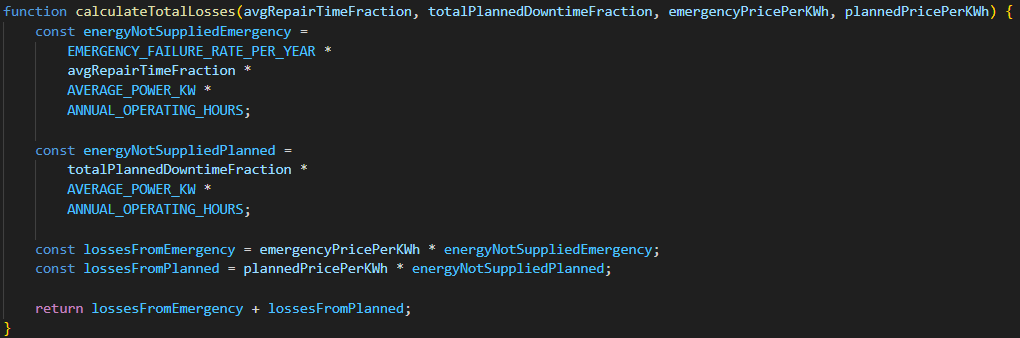
В результаті, змінні emergencyPrice,  plannedPrice,  avgRepairTime, та plannedDowntime будуть містити числові значення, введені користувачем (або 0, якщо поля були порожні чи містили некоректні дані).

**Розрахунок загальної суми збитків:**



Викликається основна функція calculateTotalLosses(), якій передаються чотири числові значення, отримані на попередньому кроці.

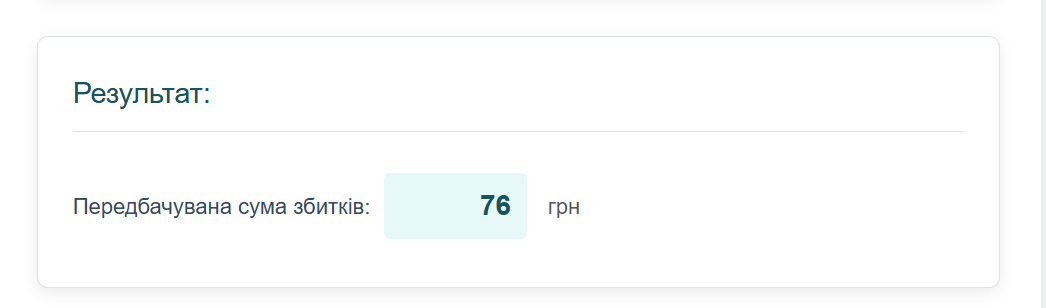
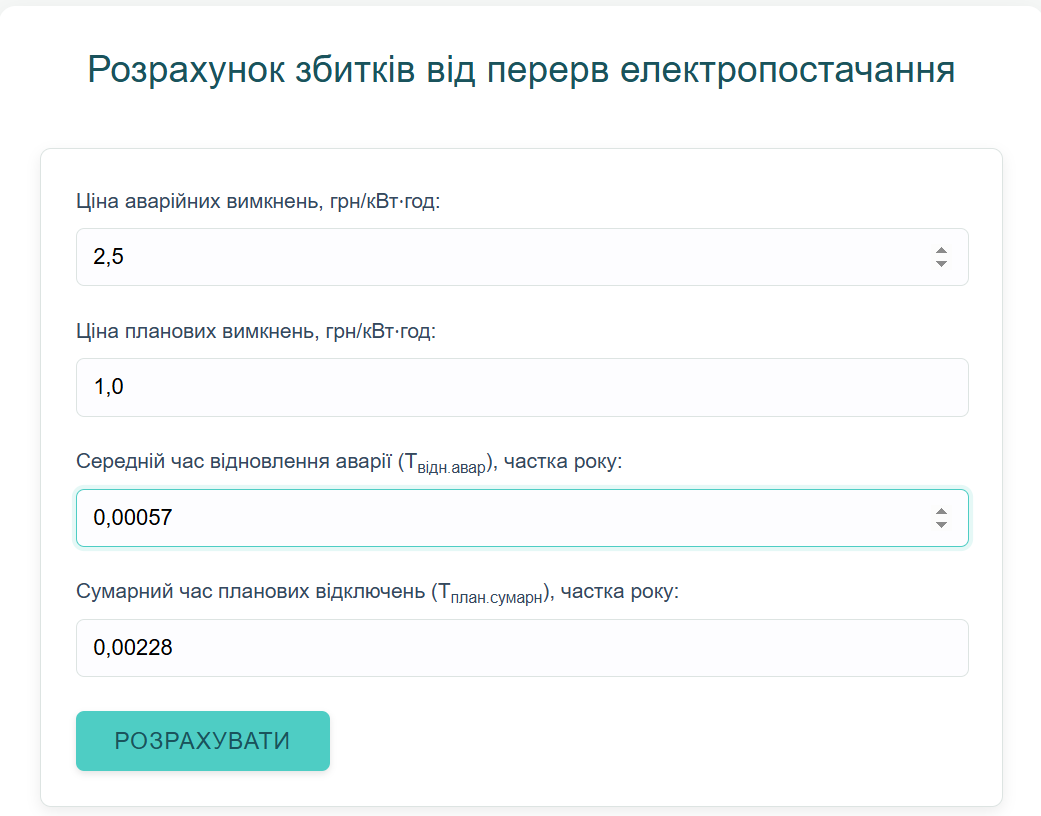
**Функція calculateTotalLosses(...):**



Це "серце" калькулятора. Вона інкапсулює всю логіку розрахунків:

1. Обчислює недовідпущену енергію через аварійні відключення (energyNotSuppliedEmergency).
2. Обчислює недовідпущену енергію через планові відключення (energyNotSuppliedPlanned).
3. Розраховує грошові збитки від аварійних відключень (lossesFromEmergency).
4. Розраховує грошові збитки від планових відключень (lossesFromPlanned).
5. Сумує ці два види збитків і повертає загальну суму.

**Результат виконання:**



**Перевірка:**

Використовуємо константи та значення за замовчуванням:

* ANNUAL\_OPERATING\_HOURS = 6451 (T)
* AVERAGE\_POWER\_KW = 5.12 (P)
* EMERGENCY\_FAILURE\_RATE\_PER\_YEAR = 0.01 (w)

Вхідні дані з полів:

* Ціна аварійних (emergencyPricePerKWh): **2.5** грн/кВт\*год
* Ціна планових (plannedPricePerKWh): **1.0** грн/кВт\*год
* Середній час відновлення аварії, частка року (avgRepairTimeFraction, з поля .t): **0.00057**
* Сумарний час планових відключень, частка року (totalPlannedDowntimeFraction, з поля .k): **0.00228**

**1. Недовідпущена енергія через аварійні відключення (M<sub>lossEm</sub>):**  
M\_lossEm = EMERGENCY\_FAILURE\_RATE\_PER\_YEAR \* avgRepairTimeFraction \* AVERAGE\_POWER\_KW \* ANNUAL\_OPERATING\_HOURS  
M\_lossEm = 0.01 \* 0.00057 \* 5.12 \* 6451  
M\_lossEm = 0.0000057 \* 33029.12  
M\_lossEm ≈ 0.188265984 кВт\*год

**2. Недовідпущена енергія через планові відключення (M<sub>lossPl</sub>):**  
M\_lossPl = totalPlannedDowntimeFraction \* AVERAGE\_POWER\_KW \* ANNUAL\_OPERATING\_HOURS  
M\_lossPl = 0.00228 \* 5.12 \* 6451  
M\_lossPl = 0.00228 \* 33029.12  
M\_lossPl ≈ 75.3063936 кВт\*год

**3. Збитки від аварійних:**  
Losses\_Emergency = emergencyPricePerKWh \* M\_lossEm  
Losses\_Emergency = 2.5 \* 0.188265984 ≈ 0.47066496 грн

**4. Збитки від планових:**  
Losses\_Planned = plannedPricePerKWh \* M\_lossPl  
Losses\_Planned = 1.0 \* 75.3063936 = 75.3063936 грн

**5. Загальна сума збитків:**  
Total\_Loss = Losses\_Emergency + Losses\_Planned  
Total\_Loss ≈ 0.47066496 + 75.3063936 ≈ 75.77705856 грн

Калькулятор округлює результат до цілого: Total\_Loss.toFixed(0) -> **76 грн**.

**Висновок:**

В ході виконання цієї лабораторної роботи я успішно розробила два інтерактивні веб-калькулятори: для аналізу надійності систем електропередачі та для розрахунку економічних збитків від перерв електропостачання. Основну увагу було приділено рефакторингу JavaScript коду для підвищення його читабельності та відповідності стандартам, а також створенню інтуїтивно зрозумілих та візуально привабливих користувацьких інтерфейсів за допомогою HTML та CSS. Я реалізувала логіку збору даних, їх обробки згідно з наданими формулами та динамічного відображення результатів.

Для забезпечення коректності розрахунків було проведено ручне тестування з використанням прикладів, яке підтвердило правильність роботи обох калькуляторів. Отриманий досвід включає практичне застосування DOM-маніпуляцій, обробки подій, роботи з формами та адаптивної верстки. Ця робота дозволила мені поглибити знання з веб-розробки та отримати навички створення функціональних веб-додатків.