Веб додаток для ефективного розміщення блоків

Завдання: розробити функціонал, який ефективно розміщує прямокутні блоки в прямокутному 2D контейнері. Розташування блоків у контейнері повинно бути якомога щільним з метою раціонального використання простору контейнера.

Завдання можно умовно розділита на 2 частини:

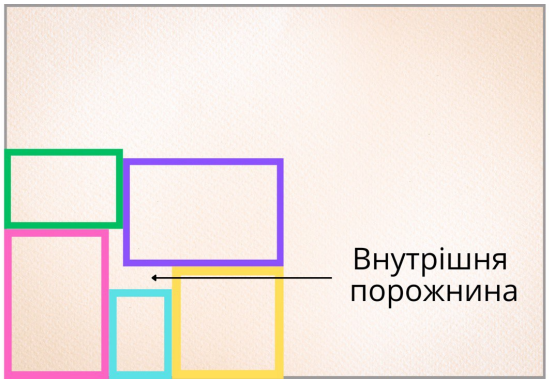
1. Створення алгоритму ефективного розміщення прямокутних блоків
2. Створення UI для відображення результату роботи алгоритма

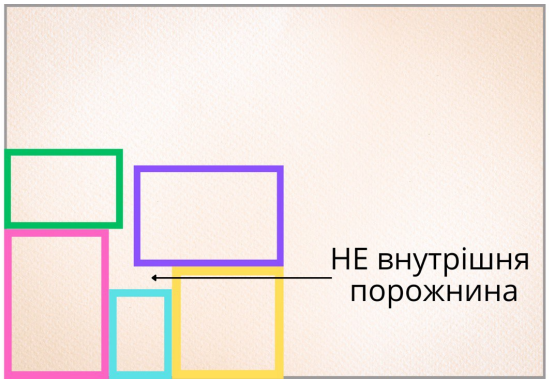
1. Алгоритм ефективного розміщення прямокутних блоків:

Створити алгоритм, який визначає найоптимальніший порядок розташування прямокутних блоків у прямокутному 2D контейнері, при якому блоки матимуть найбільш щільне розміщення та витрачатимуть найменший простір контейнера. Блоки можна провертати на 90°. Блоки не повинні перекривати один одного. Алгоритм повинен розраховувати коефіцієнт корисного використання простору зайнятого контейнерами (*fullness*), який можна виразити наступною формулою:

$$1 - \left(\frac{\text{площа внутрішніх порожнин}}{\text{площа внутрішніх порожнин} + \text{площа всіх блоків}}\right)$$

Внутрішня порожнина - це простір повністю оточений блоками (на 100%), що залишається незаповненим внаслідок не щільного розташування блоків. Простір що не повністю оточений блоками не є внутрішньою порожниною.





Вхідні параметри:

1. Список параметрів прямокутних блоків із зазначенням їх ширини та висоти. Список параметрів блоків повинен знаходитися в окремому JSON файлі. Приклад списка параметрів блоків:

```
[{ width: 90, height: 90 }, { width: 60, height: 115 }, ...]
```

2. Розмір контейнера із зазначенням його ширини та висоти, наприклад:

```
{ width: 350, height: 300 }
```

Вихідні дані:
Об'єкт з коефіцієнтом корисного простору (*fullness*), та список з координатами найефективнішого розміщення блоків всередині контейнера (*blockCoordinates*):

```
{
 fullness: 1,
 blockCoordinates: [
   { top: 400, left: 0, right: 200, bottom: 600, initialOrder: 2 },
   { top: 300, left: 0, right: 300, bottom: 400, initialOrder: 6 },
    ...
 ],
};
```

2. UI для відображення результату роботи алгоритма

Створити UI який буде відображати результат роботи алгоритму на сторінці у браузері. На сторінці повинні відображатись блоки у контейнері. Використовувати розмір viewport (його ширину та висоту) як розмір контейнера.

Вимоги до UI:

- Кожен блок має бути унікального кольору, окрім блоків однакового розміру (у них повинен бути однаковий колір)
- Кожен блок в середині має відображати свій первісний порядковий номер (індекс елемента масиву).
- При зміні розміру контейнера (viewport), алгоритм має автоматично перераховувати розташування блоків і відображати на сторінці оновлений результат.
- Відобразити коефіцієнт корисного використання простору (зверху)

Приклад того як може виглядати UI наведено на наступній сторінці.

Важливо!

1. Завдання повинно бути виконано БЕЗ використання будь яких бібліотек або фреймворків.
2. Для реалізації задачі не потрібно створювати бекенд. Достатньо, щоб вся логіка виконувалася на клієнтській частині (в браузері).
3. Завдання повинно бути виконано на JavaScript або TypeScript.

4. Завдання необхідно виконати у повному обсязі з урахуванням усіх наведених вимог.
5. Код повинен бути структурованим, розділеним на логічні блоки, компактним та легко читатись.
6. Завдання повинно бути вирішено оптимальним чином і потребувати мінімальну кількість ресурсів для його роботи.
7. Додаток повинен коректно працювати з широким спектром вхідних даних, а також коректно обробляти виняткові ситуації.
8. Результат виконаного завдання прислати zip архівом на адресу recruiter@reasonway.com

Приклад як може виглядати UI: