ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ВАСИЛЯ СТЕФАНИКА

ФАКУЛЬТЕТ МАТЕМАТИКИ ТА ІНФОРМАТИКИ

КАФЕДРА ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ І ПРИКЛАДНОЇ МАТЕМАТИКИ

**З В І Т**

з виробничої практики

студентки 4 курсу освітнього рівня бакалавр групи ПМ-41

Шинкар Анни Михайлівни

База практики ТзОВ Елекс

Термін практики з «30» січня 2023 р.

до «10» березня 2023 р.

Керівники практики:

Від бази директор Коновалов П. О.

Від кафедри доцент Мазуренко В. В.

ЗМІСТ

[ВСТУП 3](#_Toc126946398)

[РОЗДІЛ 1. Актуальність Front-end розробки 4](#_Toc126946399)

[***1.1.*** ***Призначення Front-end розробки*** 4](#_Toc126946400)

[***1.2.*** ***Компоненти Front-end розробки*** 4](#_Toc126946401)

[РОЗДІЛ 2. Розробка браузерного варіанту гри «Пятнашки» 6](#_Toc126946402)

[***2.1. Опис гри*** 6](#_Toc126946403)

[***2.2. Web-верстка*** 6](#_Toc126946404)

[***2.3. Стилізація CSS*** 7](#_Toc126946405)

[***2.4. Анімація гри JavaScript*** 9](#_Toc126946406)

[ВИСНОВКИ 12](#_Toc126946407)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 13](#_Toc126946408)

[ДОДАТОК 1 14](#_Toc126946409)

# ВСТУП

Виробнича практика є невід’ємною складовою частиною процесу підготовки фахівців з вищою освітою різних спеціальностей. Вона є одним з важливих етапів формування кваліфікованого спеціаліста, здатного самостійно вирішувати конкретні завдання в діяльності установ та організацій різних форм власності.

Метою виробничої практики є формування у студентів, на базі здобутих під час навчання знань, професійних умінь і навичок для прийняття самостійних рішень під час конкретної роботи в реальних умовах, оволодіння ними сучасними методами, формами організації та знаряддями праці, виховання потреби систематично поновлювати свої знання та творчо застосовувати їх у практичній діяльності.

Мені випала можливість пройти виробничу практику на базі інноваційної компанії Eleks. Тривалість практики, згідно наказу, у період з 30 січня по 10 березня.

ELEKS – інженерна інноваційна компанія, яка надає послуги з розробки комплексних програмних продуктів для великих гравців світового ринку в галузях фінансів, страхування, роздрібної торгівлі, транспортування, логістики та інших, серед яких компанії з рейтингу Fortune 500. Понад 900 працівників працюють у Львові, Києві, Івано-Франківську, Тернополі та Жешові, де розміщені центри розробки. З 2014 року ELEKS входить до списку топ-100 найкращих аутсорсингових компаній світу (Global Outsourcing 100) за версією міжнародної асоціації аусорсинг-професіоналів (International Association of Outsourcing Professionals). Компанія також відома успішними R&D-експериментами.

**Мета практики**: вдосконалити набуті знання у галузі веб-розробки, використати їх на практиці виконуючи індивідуальне завдання; навчитись співпрацювати з колективом на обраній базі практики й виконувати поставлені завдання керівником практики.

**Завдання практики**: вивчення організації і етапів розробки програмного продукту, набуття практичних навичок програмування, проектування та реалізації web-проектів; створення браузерного варіанту гри “Пятнашки”.

# РОЗДІЛ 1. Актуальність Front-end розробки

## ***Призначення Front-end розробки***

Front-end – це публічна частина web-додатків (веб-сайтів), з якою користувач може взаємодіяти і контактувати напряму. У Front-end входить відображення функціональних завдань призначеного для користувача інтерфейсу, що виконуються на стороні клієнта, а також обробка запитів користувачів. По суті, фронтенд – це все те, що бачить користувач при відкритті web-сторінки.

У свою чергу, web-додаток – клієнт-серверний додаток, в якому клієнтом виступає в основному браузер, а сервером – web-сервер. Логіка web-додатку розподілена між сервером і клієнтом, зберігання даних здійснюється переважно на сервері, обмін інформацією відбувається у мережі. Простіше кажучи, це те, що бачить користувач і які дії виконує кожен раз, коли підключається до мережі інтернет і відкриває будь-який браузер.

Front-end розробка – це робота зі створення публічної частини web-додатку, з якою безпосередньо контактує користувач, і функціоналу, який зазвичай виконується на стороні клієнта. Тобто, фронтенд розробник працює над тим, щоб на сайті кожна кнопочка, іконка, текст і вікно не тільки стояли на своєму місці, не перекривали один одного і виглядали цілісно (це веб-верстка), але і щоб вони виконували своє пряме призначення – підштовхували до якоїсь дії (наприклад, щоб кнопка “купити” відкривала кошик, а “play” – запускала відтворення фільму або музики).

## ***Компоненти Front-end розробки***

Основними компонентами при Front-end розробці є: HTML (HyperText Markup Language), CSS (Cascading Style Sheets), JavaScript.

***HTML (Hypertext Markup Language — Мова гіпертекстової розмітки)*** — це мова опису структури сторінок документів, яка дозволяє звичайний текст форматувати в абзаци, заголовки, списки та інші структури, створювати посилання на інші сторінки. Це текстова мова, в якій інструкції з форматування, що називаються тегами, вбудовані в розділи документа, які містять конкретну інформацію. Теги повідомляють браузерам, як форматувати і представляти інформацію на екрані.

Мова НТМL дозволяє визначити структуру електронного документа з поліграфічним рівнем оформлення. Результуючий документ може містити різноманітні елементи: ілюстрації, аудіо і відео фрагменти. Мова НТМL включає розвинені засоби для визначення кількох рівнів заголовків, шрифтових виділень, різних груп об’єктів та багато інших можливостей.

***CSS (Cascading Style Sheets або каскадні таблиці стилів)*** називається мова візуального представлення вмісту HTML-документа. Він має на увазі додавання кольору, редагування відступів, шрифтів, розташування кожного блоку на сторінці.

HTML організовує інформацію і надає структуру документу, а CSS оформляє його, прописуючи унікальні властивості кожному з елементів.

CSS – це файл, в якому прописується правила оформлення всіх елементів на сторінці. Завдяки цьому код містить менше повторюваних елементів, його простіше читати. Можна поширити одну таблицю стилів на безліч сторінок, істотно знизивши час на верстку.

Кожне правило в файлі складається з селектора і блоку оформлення. Функція селектора -визначити, до якої саме частини документа застосувати конкретне правило. Блок оформлення складається з пар «властивість: значення». Вони пишуться в фігурних дужках і закінчуються крапкою з комою.

***JavaScript*** – це мова програмування, що дозволяє зробити Web -сторінку інтерактивною, тобто такою що реагує на дії користувача.

Послідовність інструкцій (що називається програмою, скриптом або сценарієм) виконується інтерпретатором, вбудованим в звичайний Web -браузер. Іншими словами, код програми вбудовується в HTML - документ і виконується на боці клієнта. Для виконання програми не потрібно навіть перезавантажувати Web -сторінку, всі програми виконуються в відповідь на будь-яку подію.

JavaScript - об'єктно-орієнтована скриптова мова програмування і є діалектом мови ECMAScript.

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить у браузерах як мова сценаріїв для надання інтерактивності веб-сторінкам.

# РОЗДІЛ 2. Розробка браузерного варіанту гри «Пятнашки»

## ***2.1. Опис гри***

**П'ятна́шки** (п'ятнадцять) — популярна головоломка, придумана у 1878 році Ноєм Чепменом. Складається з 15 однакових квадратних пластинок з нанесеними числами від 1 до 15. Пластинки поміщаються в квадратну коробку, довжина сторони якої в чотири рази більша довжини сторони пластинок, відповідно в коробці залишається незаповненим одне квадратне поле (рис. 1).

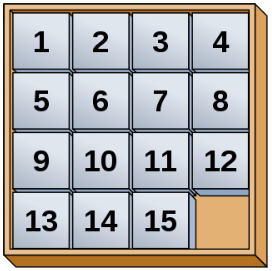


Рисунок 1

Мета гри — переміщаючи пластинки по коробці добитися впорядковування їх по номерах (як зображено на рисунку), бажано зробивши якомога менше переміщень.

Позначивши пусту клітину числом 16, властивості головоломки можна одержувати із властивостей перестановок чисел 1..16. Так кожне переміщення у головоломці відповідає деякій транспозиції числа 16 і деякого іншого числа. Кожна така транспозиція змінює парність перестановки чисел.

Якщо в початковому розміщенні пуста клітина розташована у нижньому правому куті, то на своє місце вона може повернутися за парну кількість кроків, тобто парність у цьому випадку не зміниться. Тому якщо початкова перестановка є непарною то головоломка розв'язку не має.

## ***2.2. Web-верстка***

Написання програми для гри буде виконуватись в Sublime Text 3 — швидкий кросплатформенний текстовий редактор. Підтримує плагіни, розроблені за допомогою мови програмування Python.

Першими рядками коду при створенні сторінки html-документа є так званий «скелет» документа через конструкцію <!doctype html >. Розглянемо рисунок 2:



Рисунок 2

Як видно з рис. 2, скелет документу складається з шапки (<head> …</head>) та тіла (<body>...</body>). У шапці за допомогою тега <title>… </title> вказується, як буде називатись відкрита в браузері сторінка. За допомогою тега <link> як видно на рис. 2 підключаємо файл, де будуть прописані стилі, для оформлення сторінки.

В тілі документа будуть розміщуватися елементи самої гри, зокрема комірки з цифрами, кнопка для того щоб розпочати гру, кількість кроків та час. Каркас гри без стилізації буде виглядати наступним чином ( див. рис. 3):



Рисунок 3

Комірки та кнопка «Грати» задані за допомогою тегу <button>…</button>, а кількість кроків та час за допомогою тегу <span>… </span>.

## ***2.3. Стилізація CSS***

Для того щоб стилізувати сторінку було створено файл style.css, в якому і прописано загальний вигляд сторінки.

Для тегу <body> були прописані наступні властивості: лінійний градієнт, елементи вирівнюються по центру сторінки, елементи відображаються у вигляді стовпчика.

Щоб взяти гру у рамку й прописати її оформлення створено клас .game, для нього задано тінь, товщину, колір та округлення всіх кутів.

Для зовнішнього вигляду комірок з цифрами створено клас .grid, в ньому задано розмір клітинок 80px на 80px, товщину та колір рамки, колір комірки, розмір і товщину шрифту, а також зміну вигляду курсору при наведенні на комірку.

Під комірками, котрі позначені цифрами, знаходиться кнопка для початку гри та поля для підрахунку часу й кількості кроків. Для цього створено клас footer. Кнопка «Грати» має ширину 80px, фон жовтого кольору та напис чорного кольору розміром 20px, а при наведенні курсору на неї колір фону стає чорним, а напис жовтим. Для такої зміни був застосований псевдоклас hover. Поля де підраховується кількість кроків та час в секундах стилізовані наступним чином: вирівнювання по центру, розмір тексту 20px, жирний, колір жовтий.

Стилі для повідомлення про виграш прописані в класі message: колір червний, висота 80px.

В файл index.html підключено файл style.css як це показано на рисунку 2. Для того, щоб відповідні елементи мали такий вигляд, який ми прописали в style.css, їх необхідно застосувати до цих елементів наступним чином: <div class="назва">. Наприклад:

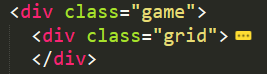


Рисунок 4

На рис. 4 показано як підключено клас game та grid.

Підключивши всі прописані класи до відповідних елементів сторінки отримуємо в результаті (рис.5):

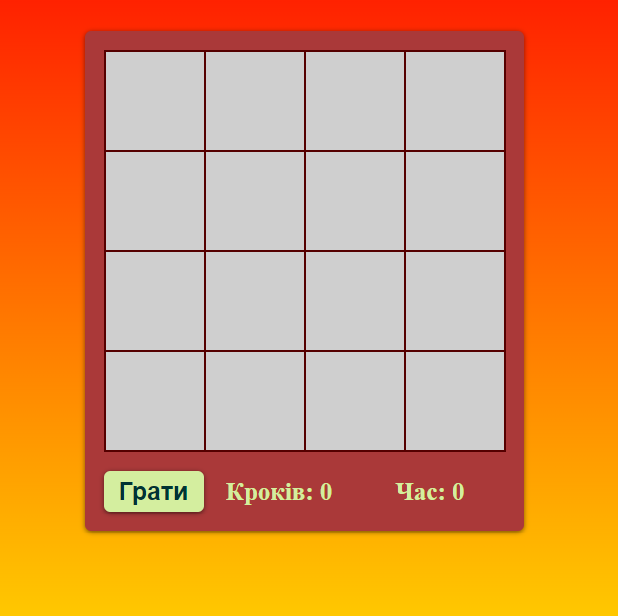


Рисунок 5

При наведенні курсору на кнопку «Грати», вона повинна змінювати колір фону й тексту (рис. 6)

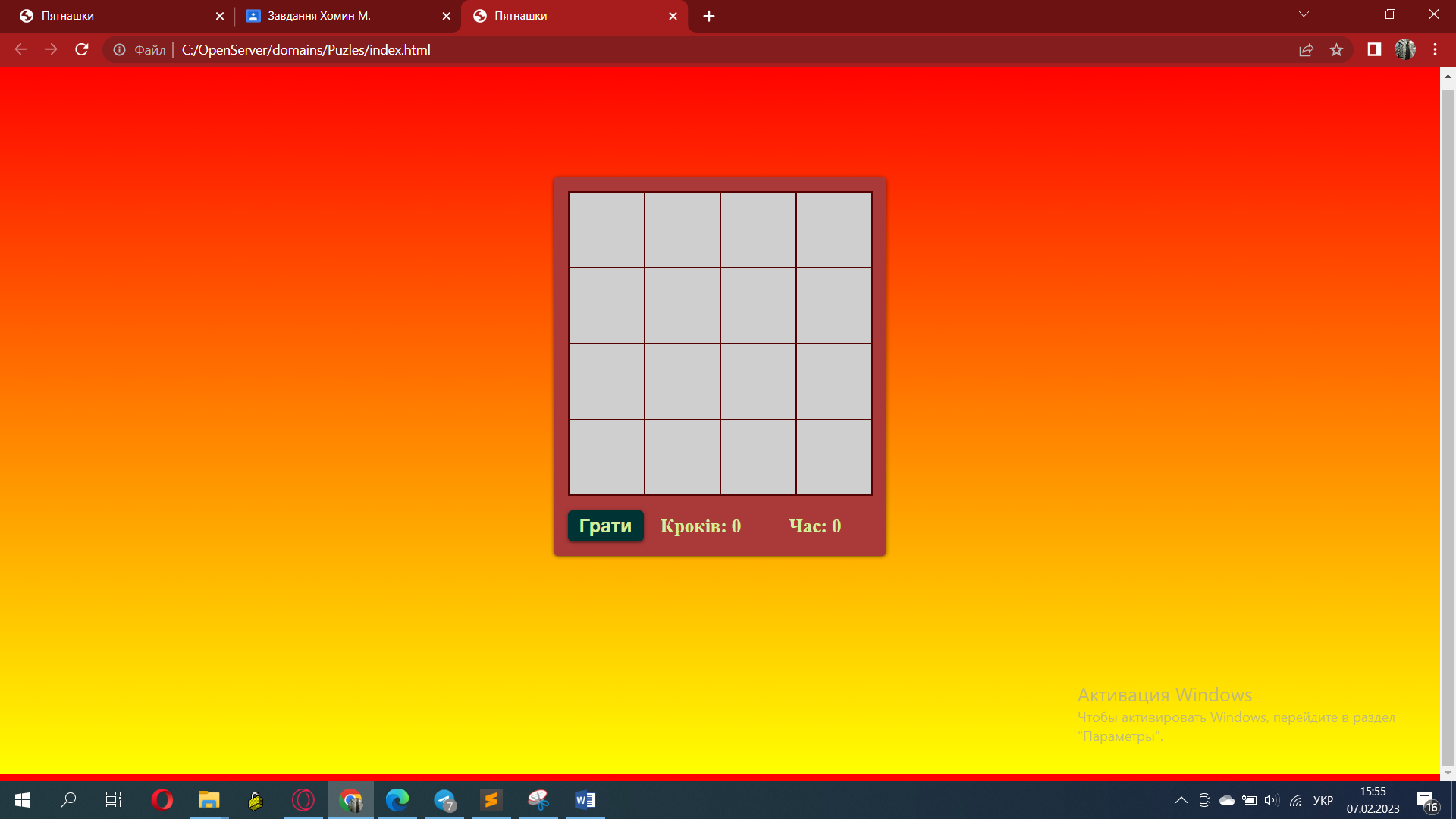


Рисунок 6

## ***2.4. Анімація гри JavaScript***

Перед тим як почнеться гра комірки, які мають бути пронумеровані цифрами є пустими, а кількість кроків та час дорівнюють нулю, як це показана на рис. 6. Коли гравець натисне на кнопку «Грати», комірки рандомно заповняться числами від 1 до 15, а одна комірка залишиться пустою. Комірки повинні заповнятися саме рандомно, відповідно до умови індивідуального завдання. Для цього було використано функцію Math.random, але так як дана функція генерує числа випадковим чином в діапазоні [0;1), а нам в клітинах потрібці цілі числа, до неї було застосовано функцію Math.floor. Результат наведено на рис. 7

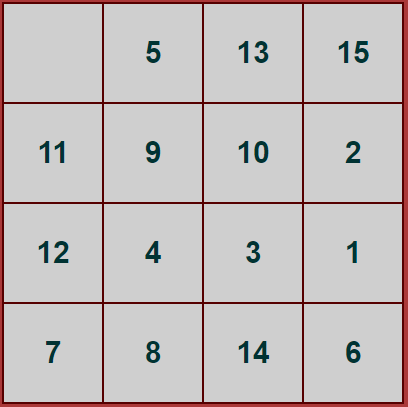
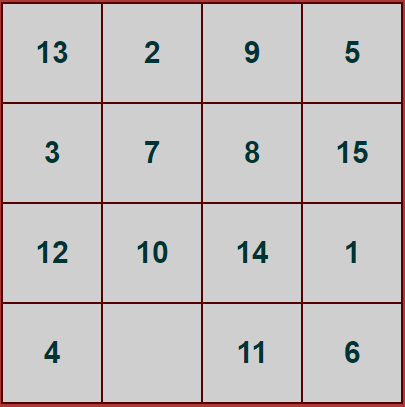
 

Рисунок 7

Для того, щоб переміщувати пронумеровані клітини на місце пустої, були створена функція render, в якій при здійсненні кліку на комірку з числом також іде перевірка чи переміщення обраної комірки не спричинило виграш, дана перевірка прописана у функції handleClickBox, якщо статус гри не дорівнює "won", то гра продовжується. Справність пересування комірок з числами наведено на рисунку 8.

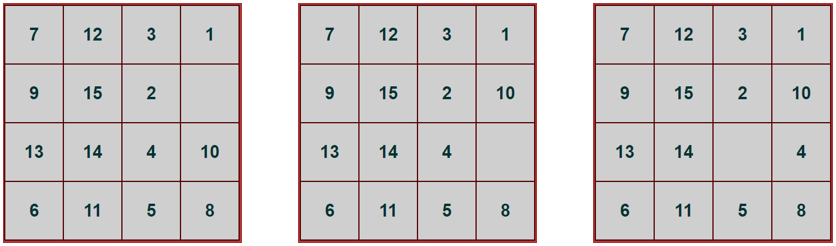


Рисунок 8

Також у функції здійснюється підрахунок переміщень комірок, тобто кількість ходів і відбувається він відповідно до наступного запису (рис. 9):

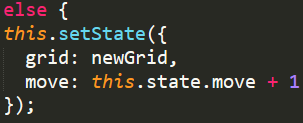


Рисунок 9

Після останнього кліку на комірку змінна grid приймає значення newGrid, бо змінилося положення комірок з числами, й збільшує кількість кроків на одиницю, нове значення передається в змінну move.

Для того щоб створити секундомір було створено функцію tick, котра збільшується на 1 секунду, поки гру не буде пройдено. Для зупинки секундоміра використано функцію clearInterval.

Щоб кнопка «Грати» при початку гри змінилась на кнопку «Нова гра» й аналогічно при виграші знову змінилась на «Грати» було прописано наступний код (рис. 10):

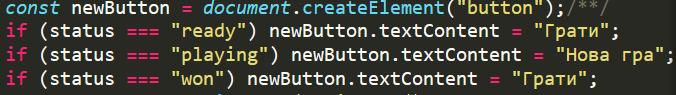


Рисунок 10

Якщо гру ще не починали то статус має значення "ready" і напис на кнопці - "Грати"; коли користувач почав гру, то статус змінюється на "playing" і відповідно напис на кнопці – «Нова гра»; коли гра пройдена статус змінюється на "won" і напис на кнопці знову змінюється на «Грати». Дана зміна відбувається завдяки властивості textContent.

Гравець під час гри може бачити як йде секундомір та як змінюється кількість кроків. Таке виведення даних реалізується наступним записом (рис.11):

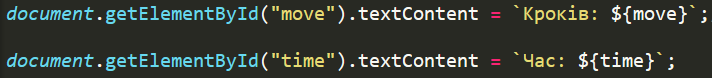


Рисунок 11

Після того як користувач пройшов гру й розмістив комірки в потрібному порядку, йому виводиться повідомлення про перемогу, під самою грою. Дана дія відбувається, коли статус гри дорівнює "won". Аналогічно до часу та кількості кроків повідомлення про перемогу передається завдяки властивості textContent. У файлі index.html був заданий заголовок під грою, з класом "message" стилі для якого ми прописували в документі style.css і реалізується дана зміна наступним записом (рис. 12):

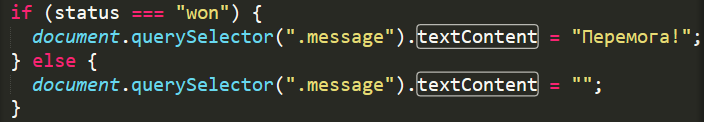


Рисунок 12

Отже, коли гра пройдена сторінка набуде наступного вигляду (рис. 13):



Рисунок 13

# ВИСНОВКИ

Під час проходження практики вдалося поглибити свої знання в сфері веб-програмування, а також застосувати їх на практиці під час створення браузерного варіанту гри «Пятнашки».

Спочатку було створено скелет майбутньої гри, де створені основні компоненти гри, які надалі стилізувались за допомогою стилів css.

Для функціонування гри була використана мова JavaScript, внаслідок чого вдалось поглиби свої знання даної мови програмування.

Відповідно до поставленого завдання було реалізовано:

* Пересування фігури, при натисканні на неї мишкою;
* Розміщення фігур випадковим чином при кожній новій грі;
* Створення кнопки для початку нової гри (без перезавантаження сторінки);
* Заборонено недозволені ходи;
* Додано секундомір, який в секундах фіксує час виграшу.

Також було прийнято рішення додати ще підрахунок кількості кроків, за допомогою якого, гравець може бачити скільки кроків йому знадобилось для перемоги.

Внаслідок зробленої роботи можна зробити висновок, що веб-програмування завжди актуальне та потрібне, а спеціалісти, які вміють розробляти веб-сайти та ігри потрібні завжди, адже технології розвиваються і рано чи пізно всі компанії стикаються з проблемою створення веб-сайту та його наповнення.

Завдяки базі практики було вдосконалено навички роботи у команді, вміння чітко й швидко виконувати поставлені завдання та креативно мислити.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лабберс П. HTML 5 для профессионалов: уч. пос. Москва: Вильямс, 2011 р., - 272с.
2. Робин Никсон. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5, 2019 р., - 816 с.
3. Дэвид Скляр. Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов, 2017 р. – 464 с.
4. Роббинс Дж. Н. HTML5, CSS3 и JavaScript : уч. пос. Москва: Эскимо, 2014. - 346 с.

# ДОДАТОК 1

***index.html***

<!doctype html>

<html>

<head>

<title>Пятнашки</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<div class="game">

<div class="grid">

<button>1</button>

<button>2</button>

<button>3</button>

<button>4</button>

<button>5</button>

<button>6</button>

<button>7</button>

<button>8</button>

<button>9</button>

<button>10</button>

<button>11</button>

<button>12</button>

<button>13</button>

<button>14</button>

<button>15</button>

<button></button>

</div>

<div class="footer">

<button>Грати</button>

<span id="move">Кроків: 100</span>

<span id="time">Час: 100</span>

</div>

</div>

<h1 class="message">Перемога!</h1>

<script src="script.js"></script>

</body>

</html>

***style.css***

html {

height: 100%;

}

body {

height: 100%;

background-image: linear-gradient(#FF0000, #FFFF00);

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

}

.game {

box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5);

padding: 15px;

background-color: #AA3939;

border-radius: 5px;

}

.grid {

display: grid;

grid-template-columns: 80px 80px 80px 80px;

grid-template-rows: 80px 80px 80px 80px;

border: 1px solid #550000;

}

.grid button {

background-color: #cfcfcf;

color: #003333;

font-size: 24px;

font-weight: bold;

border: 1px solid #550000;

outline: none;

cursor: pointer;

}

.footer {

margin-top: 15px;

display: flex;

justify-content: space-between;

}

.footer button {

border: none;

font-size: 20px;

font-weight: bold;

border-radius: 5px;

box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5);

padding: 5px;

width: 80px;

background-color: #D4EE9F;

color: #003333;

outline: none;

cursor: pointer;

}

.footer button:hover {

color: #D4EE9F;

background-color: #003333;

}

.footer span {

flex: 1;

text-align: center;

font-size: 20px;

color: #D4EE9F;

font-weight: bold;

margin: auto 0;

}

.message {

color:#AA3939;

height: 80px;}

***script.js***

class Box {

constructor(x, y) {

this.x = x;

this.y = y;

}

getTopBox() {

if (this.y === 0) return null;

return new Box(this.x, this.y - 1);

}

getRightBox() {

if (this.x === 3) return null;

return new Box(this.x + 1, this.y);

}

getBottomBox() {

if (this.y === 3) return null;

return new Box(this.x, this.y + 1);

}

getLeftBox() {

if (this.x === 0) return null;

return new Box(this.x - 1, this.y);

}

getNextdoorBoxes() {

return [

this.getTopBox(),

this.getRightBox(),

this.getBottomBox(),

this.getLeftBox()

].filter(box => box !== null);

}

getRandomNextdoorBox() {/\*розміщення чисел рандомно\*/

const nextdoorBoxes = this.getNextdoorBoxes();

return nextdoorBoxes[Math.floor(Math.random() \* nextdoorBoxes.length)];

}}

const swapBoxes = (grid, box1, box2) => {

const temp = grid[box1.y][box1.x];

grid[box1.y][box1.x] = grid[box2.y][box2.x];

grid[box2.y][box2.x] = temp;};

const isSolved = grid => {

return (

grid[0][0] === 1 &&

grid[0][1] === 2 &&

grid[0][2] === 3 &&

grid[0][3] === 4 &&

grid[1][0] === 5 &&

grid[1][1] === 6 &&

grid[1][2] === 7 &&

grid[1][3] === 8 &&

grid[2][0] === 9 &&

grid[2][1] === 10 &&

grid[2][2] === 11 &&

grid[2][3] === 12 &&

grid[3][0] === 13 &&

grid[3][1] === 14 &&

grid[3][2] === 15 &&

grid[3][3] === 0

); };

const getRandomGrid = () => {

let grid = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 0]];

let blankBox = new Box(3, 3);

for (let i = 0; i < 1000; i++) {

const randomNextdoorBox = blankBox.getRandomNextdoorBox();

swapBoxes(grid, blankBox, randomNextdoorBox);

blankBox = randomNextdoorBox;

}

if (isSolved(grid)) return getRandomGrid();

return grid;};

class State {

constructor(grid, move, time, status) {

this.grid = grid;

this.move = move;

this.time = time;

this.status = status;

}

static ready() {

return new State(

[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]], 0, 0, "ready"); }

static start() {

return new State(getRandomGrid(), 0, 0, "playing"); }}

class Game {

constructor(state) {

this.state = state;

this.tickId = null;

this.tick = this.tick.bind(this);

this.render();

this.handleClickBox = this.handleClickBox.bind(this);}

static ready() {

return new Game(State.ready());}

tick() {

this.setState({ time: this.state.time + 1 });}

setState(newState) {

this.state = { ...this.state, ...newState };

this.render();}

handleClickBox(box) {

return function() {

const nextdoorBoxes = box.getNextdoorBoxes();

const blankBox = nextdoorBoxes.find(

nextdoorBox => this.state.grid[nextdoorBox.y][nextdoorBox.x] === 0 );

if (blankBox) {

const newGrid = [...this.state.grid];

swapBoxes(newGrid, box, blankBox);

if (isSolved(newGrid)) {

clearInterval(this.tickId);

this.setState({

status: "won",

grid: newGrid,

move: this.state.move + 1});

} else {

this.setState({

grid: newGrid,

move: this.state.move + 1 }); }}}.bind(this);}

render() {

const { grid, move, time, status } = this.state;

const newGrid = document.createElement("div");

newGrid.className = "grid";

for (let i = 0; i < 4; i++) {

for (let j = 0; j < 4; j++) {

const button = document.createElement("button");

if (status === "playing") {

button.addEventListener("click", this.handleClickBox(new Box(j, i))); }

button.textContent = grid[i][j] === 0 ? "" : grid[i][j].toString();

newGrid.appendChild(button);}}

document.querySelector(".grid").replaceWith(newGrid);

const newButton = document.createElement("button");

if (status === "ready") newButton.textContent = "Грати";

if (status === "playing") newButton.textContent = "Нова гра";

if (status === "won") newButton.textContent = "Грати";

newButton.addEventListener("click", () => {

clearInterval(this.tickId);

this.tickId = setInterval(this.tick, 1000);

this.setState(State.start()); });

document.querySelector(".footer button").replaceWith(newButton);

document.getElementById("move").textContent = `Кроків: ${move}`;

document.getElementById("time").textContent = `Час: ${time}`;

if (status === "won") {

document.querySelector(".message").textContent = "Перемога!";

} else {

document.querySelector(".message").textContent = "";}}}

const GAME = Game.ready();