МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. СІКОРСЬКОГО»

Кафедра інформаційних систем та технологій

**МЕТОДИЧНІ ВКАЗІВКИ**

до виконання лабораторних робіт

з курсу **«****Основи FE-технологій»**

КИЇВ-2025

**Увага!** Варіанти лабораторних робіт відповідають **номеру студента за загальним списком з усіх груп (див. таблицю на Google Drive)**. При написанні кода **html**  тег **title** має містити інформацію про студента (**ПІБ, група,номер лаб. роботи, номер варіанта**).

Звіти з лаб. роб. слід надсилати через Google Form (див. посилання в Telegram групі)

# **Звіти до лабораторних робіт**

У звітах до лабораторних робіт варто описувати, які засоби/ бібліотеки/фреймворки/інструменти було використано, які саме можливості цих засобів було використано, як можна використовувати такі можливості (які команди виконуються для їх встановлення чи підключення, які функції/методи/об’єкти використовуються, які параметри передаються і так далі). Якщо для використаних засобів є альтернативи – варто коротко порівняти, описати переваги та недоліки, вказати аргументацію чому обрано саме ці засоби. Треба навести посилання на документацію (не загальне посилання на початкову сторінку, а конкретно на ті частини документації, де іде мова про використані засоби), а також можливо на інші використані ресурси.

Не треба копіювати весь написаний код у звіт (у вигляді тексту чи зображення) – код має бути доступним в репозиторії, варто додати точне посилання на нього у звіт (як каталоги, де знаходиться це завдання, так і коміти чи теги, які відповідають виконаному завданню). Можна наводити короткі фрагменти коду, якщо це сприяє розумінню звіту. Аналогічно, не треба копіювати великі фрагменти з документації – краще просто зробити посилання на них. Не варто орієнтуватись на обсяг звіту – короткий (1-2 сторінки), але зрозумілий і написаний по суті звіт краще, ніж десятки сторінок, щодо яких виникають питання, як саме вони пов’язані із завданням.

Якщо розроблений застосунок має графічний інтерфейс користувача (для більшості лабораторних з frontend це є обов’язковою вимогою) – у звіті варто додати скріншоти, що демонструють інтерфейс розробленого застосунку. Якщо застосунок розгорнутий на зовні доступному сервері – треба надати посилання (наприклад, GitHub Pages, Microsoft Azure, Heroku, AWS, …). Окрім цього, у звіті або у репозиторії мають бути чіткі інструкції щодо того, як запустити розроблений застосунок локально (особливо це стосується навчального проєкту, а також лабораторних з React чи подібних фреймворків).

У випадку виконання завдань у команді, доцільно створити один звіт від усієї команди, а не окремі звіти від окремих учасників. Окремі розділи звіту можуть складати різні студенти, що варто відмітити у звіті. Також у звіті варто описати, за що відповідали окремі учасники команди (які компоненти, обов’язки, …) . Не треба детально описувати кожен коміт, зроблений різними учасниками – ця інформація і так доступна в репозиторії. Натомість, варто описати запланований розподіл обов’язків – навіть якщо на практиці були певні зміни в цьому плані.

Звіти оформлюються відповідно до стандартів оформлення документації. Титульна сторінка містить назву дисципліни, інформацію про студентів (прізвище, ім’я, по батькові, група), номер та назву завдання, номер варіанту.

# **Лабораторна робота №1**

# **Тема: «Мова гіпертекстової розмітки сайту HTML-5»**

**Завдання.**

1. В текстовому редакторі VSCode cтворити WEB-сторінку мовою HTML-5. Обов'язково використовувати теги <TITLE> І <META>. У **TITLE вказати своє прізвище, ініціали та номер групи.** В META - кодування сторінки, keywords, description

2. Створити 3 абзаци з інформацією про автора сторінки:

1-й - прізвище, ім'я, по батькові (використовувати **тег H2**);

2-й - інформацію про дату і місце народження;

3-й - інформацію про освіту (школа, університет).

3. Інформацію про ваше хобі -у вигляді **ненумерованого списку** (не менше 3-пунктів).

4. Три ваших улюблених книги або фільми у вигляді **нумерованого списку**.

5. **Абзац** з описом самого вподобаного міста, де ви бували.

6. Додати **фото** цього міста.

7. Додати **гіперпосилання** на це фото з адресою сайту цього міста.

8. **Робота з Git.**

9. Використання **валідаторів HTML.**

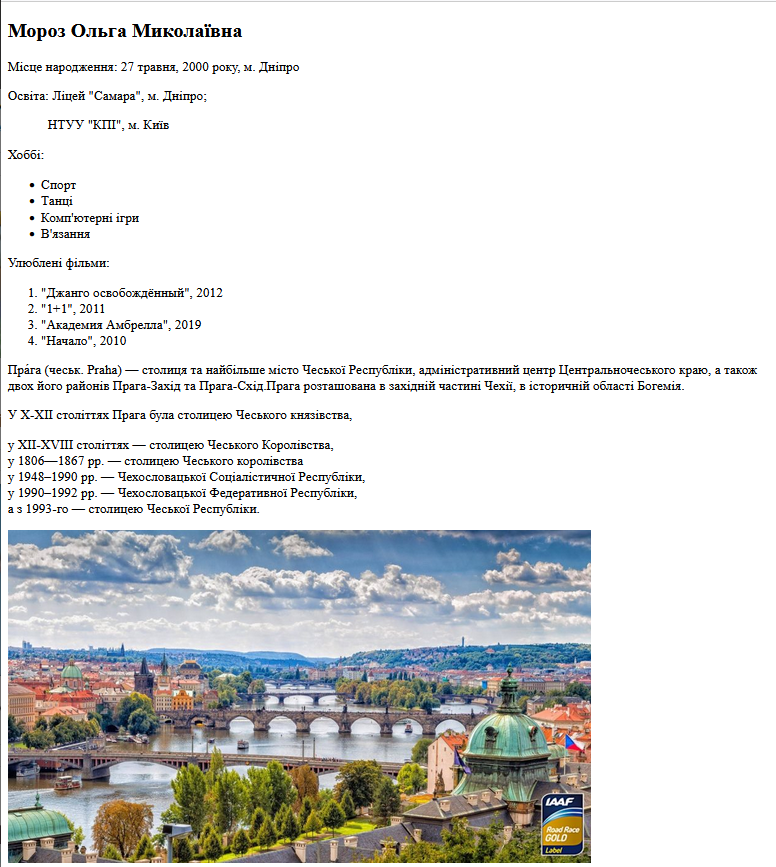
Завантажити з сайту git-scm.com інсталяцію пакета контроля версій git та встановити її на свій комп’ютер.

Зареєструватися на git-hub та створити свій репозиторій. Завантажити дані лаб. роб. в створений репозиторій.

Завантажити створену сторінку на хостінг в GitHub.

Провести валідацію створеної сторінки, використовуючи валідатори HTML (наприклад валідатор W3C <https://validator.w3.org/> ). Результати валідації навести у звіті, пояснити ці результати (зокрема знайдені недоліки).

**Приклад виконання лабораторної роботи:**



**Оформлення звіту та порядок захисту** Лабораторна робота виконується на комп’ютері. Звіт оформлюється на аркушах А4 , в якому стисло відображається зміст, хід роботи та отримані результати. **Обов’язково навести посилання на проект в Git-hub та посилання на створений сайт.**

# **Лабораторна робота №2**

# **Тема «Таблиці в HTML-5. Створення зображень засобами трансформацій в CSS-3»**

**Завдання 1.**

За допомогою мови HTML-5 створити таблицю відповідно за своїм варіантом. У перший рядок таблиці розташувати ПІБ виконавця (окремо в кожну комірку). У другому рядку записати послідовно групу, номер лабораторної роботи та номер варіанту.

В цьому ж файлі додати CSS властивості:

Для 1-го рядка встановити колір #nn8f8f та 2 #f5f5nn (властивість background-color), де nn- номер вашого варіанту.

Для тексту обрати контрастний колір за допомогою сервісу [color.adobe.com](https://color.adobe.com/)

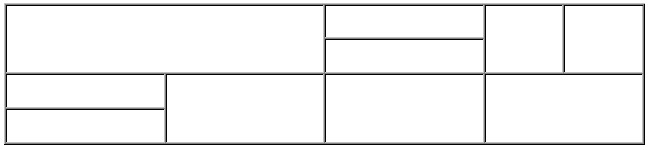
Зробити таким чином, щоб при наведенні курсора на 1-й та 2-й рядки кольори змінювалися на протилежні (використати псевдо-клас hover).

**Завдання 2.**

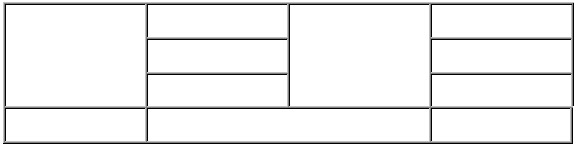
Створити зображення відповідно до свого варіанта: 1. – за допомогою position; 2- за допомогою transform , використовуючи тільки властивості CSS. Властивості CSS зберегти в окремому файлі style.css.

**Варіанти до завдання 1.**

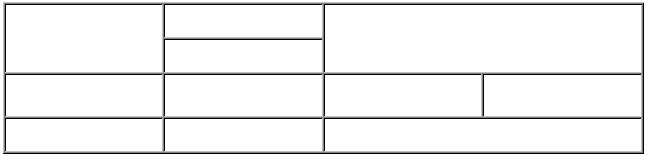
**Варіант 1**



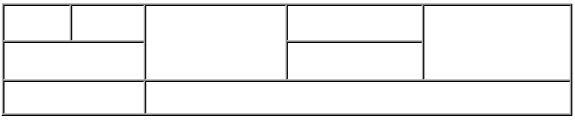
**Варіант 2**



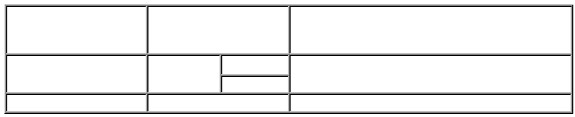
**Варіант 3**



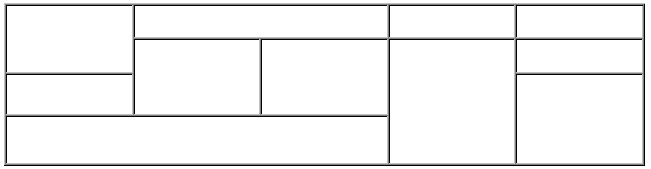
**Варіант 4**



**Варіант 5**



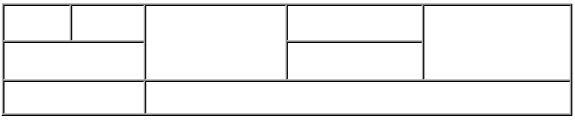
**Варіант 6**



**Варіант 7**



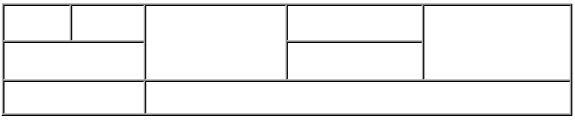
**Варіант 8**



**Варіант 9**



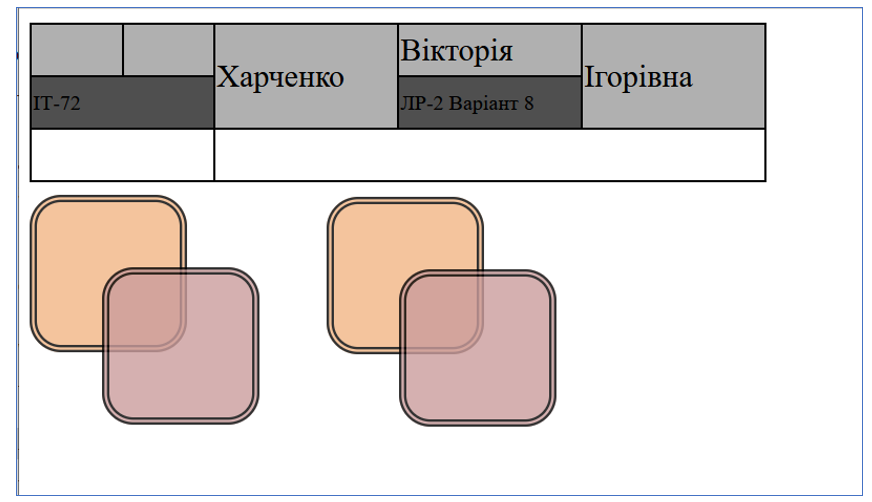
**Варіант 10**



**Варіанти до завдання 2.**

|  |  |
| --- | --- |
| **Варіант 1:** | **Варіант 6:** |
|  | 12:00 |
| **Варіант 2:** | **Варіант 7:** |
|  |  |
| **Варіант 3:** | **Варіант 8:** |
|  |  |
| **Варіант 4:** | **Варіант 9:** |
|  |  |
| **Варіант 5:** | **Варіант 10:** |
|  |  |

**Приклад виконання лабораторної роботи:**



**Оформлення звіту та порядок захисту**

**Оформлення звіту та порядок захисту** Лабораторна робота виконується на комп’ютері. Звіт оформлюється на аркушах А4 , в якому стисло відображається зміст, хід роботи та отримані результати. **Обов’язково навести посилання на проект в Git-hub та посилання на створений сайт.**

**Теоретичні відомості**

**Робота з таблицями.**

Сама таблиця має жорстку структуру: головним є тег <table>, усередині якого повинні лежати теги <tr>, які створюють ряди (рядки) таблиці, а всередині них - теги <td>, які створюють осередки.

Як ви бачите, немає тегів, які створювали б стовпці - таблиці створюються по рядах: спочатку перший ряд, потім другий і так далі.

<table border="1">

<!--Це перший ряд таблици:-->

<tr>

<td>Ячейка 1</td>

<td>Ячейка 2</td>

<td>Ячейка 3</td>

</tr>

<!--Це другий ряд таблици:-->

<tr>

<td>Ячейка 4</td>

<td>Ячейка 5</td>

<td>Ячейка 6</td>

</tr>

<!--Це третій ряд таблици:-->

<tr>

<td>Ячейка 7</td>

<td>Ячейка 8</td>

<td>Ячейка 9</td>

</tr>

</table>

**Розширимо Ячейку1 на два стовпці без розвалювання таблиці**

Щоб поправити проблему з розвалювання таблиці з попереднього прикладу, видалимо одну з комірок праворуч від нашої (це Ячейка2 або Ячейка3 - без різниці):

<table>

<tr>

<td colspan="2">Ячейка 1</td>

<td>Ячейка 2</td>

</tr>

<tr>

<td>Ячейка 4</td>

<td>Ячейка 5</td>

<td>Ячейка 6</td>

</tr>

<tr>

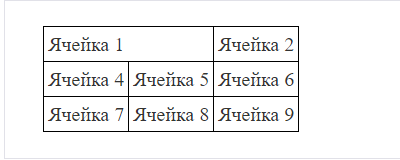
<td>Ячейка 7</td>

<td>Ячейка 8</td>

<td>Ячейка 9</td>

</tr>

</table>



Каскадні таблиці стилів CSS (Cascading style sheets) - формальна мова опису зовнішнього вигляду документа, створеного з використанням мови розмітки гіпертексту.

Каскадні таблиці стилів дозволяють розділити опис логічної структури html-документа (виконане за допомогою мови розмітки) і опис зовнішнього вигляду html-документа (виконане за допомогою CSS).

Існує три способи визначення стилів: 1) в окремому файлі, що підключається до html-документів, 2) за допомогою тега <style> безпосередньо в деякому html-документі і 3) за допомогою атрибута style безпосередньо в деякому тезі.

Найбільш високий пріоритет має стиль, визначений у тезі, потім йде визначення стилю за допомогою тега style і найнижчим пріоритетом мають властивості, певні в окремому файлі.

Каскад пріоритетів особливо зручний при розробці великих проектів, наприклад, сайтів, які з значної частини html-документів. У цьому випадку загальне оформлення може бути винесено в окремий файл, в html-документі можуть бути внесені зміни в стиль документа за допомогою тега <style>, атрибут тега style дозволяє змінити оформлення одного тега.

Стилі визначаються парами властивостей і значень, перелік пар полягає в фігурні дужки і пари розділяються крапкою з комою:

{Property\_1: value\_1; property\_2: value\_2; ...; property\_n: value\_n}

**Абсолютне позиціонування**

Спільно з властивістю position використовуються CSS властивості, які керують зміщенням позиціонованого елемента:

• top (зміщення позиціонованого елемента від верхнього краю).

• right (зміщення позиціонованого елемента від правого краю).

• bottom (зміщення позиціонованого елемента від нижнього краю).

• left (зміщення позиціонованого елемента від лівого краю).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Як значення, які визначають зміщення елемента, допускається використовувати фізичні одиниці (наприклад, пункти ), але частіше використовують візуальні одиниці - пікселі , процентні значення і значення em . Значення можуть бути як позитивні , так і негативні , як і люди, тільки значення.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

При використанні абсолютного позиціонування ( **position : absolute** ) елемент зсувається (позиціонується) щодо заданого краю його предка , при цьому предок повинен мати значення position відмінне від, встановленого за замовчуванням - static , інакше відлік (зміщення) буде вестися щодо, зазначеного краю вікна браузера .

**Відносне позиціонування**

Наступний тип позиціонування, який ми розглянемо це відносне позиціонування . Елементи, для яких задано відносне позиціонування ( [position](https://basicweb.ru/css/css_pr_position.php) : relative ) зміщуються (розміщуються) щодо положення в потоці документа , або іншими словами щодо його поточної позиції . Основна ідея відносного позиціонування полягає в тому, щоб не зрушити кудись елемент, а створити «контейнер» для елемента, який має абсолютне позиціонування . Іншими словами, вкладені елементи будуть зміщуватися не відносно краю вікна браузера, а щодо цього елемента, який буде мати відносне позиціонування і знаходиться в основному потоці документа .

# **Лабораторна робота №3**

# **Тема: «Блочна верстка макету сайту за допомогою HTML і CSS»**

Завдання 1.

Макет структури сайту виконано в PhotoShop. За допомогою **HTML5** та **CSS3** зробити блочну верстку розробленогомакету використовуючи технологію **float**. Обов’язково використати **width** у відсотках.

**Завдання 2.**

Створити верстку цього ж макету сайту за допомогою технології **Flex.**

|  |  |
| --- | --- |
| Варіанти: |  |
| 1 | Рисунок 77 |
| 2 | Рисунок 78 |
| 3 | Рисунок 79 |
| 4 | Рисунок 80 |
| 5 | Рисунок 81 |
| 6 | Рисунок 82 |
| 7. | Рисунок 83 |
| 8. | Рисунок 121 |
| 9. | Рисунок 152 |
| 10. | Рисунок 153 |

**Теоретична інформація**

Взаємодія між графічним дизайнером та frontend розробником (зокрема верстальником) зазвичай відбувається наступним чином:

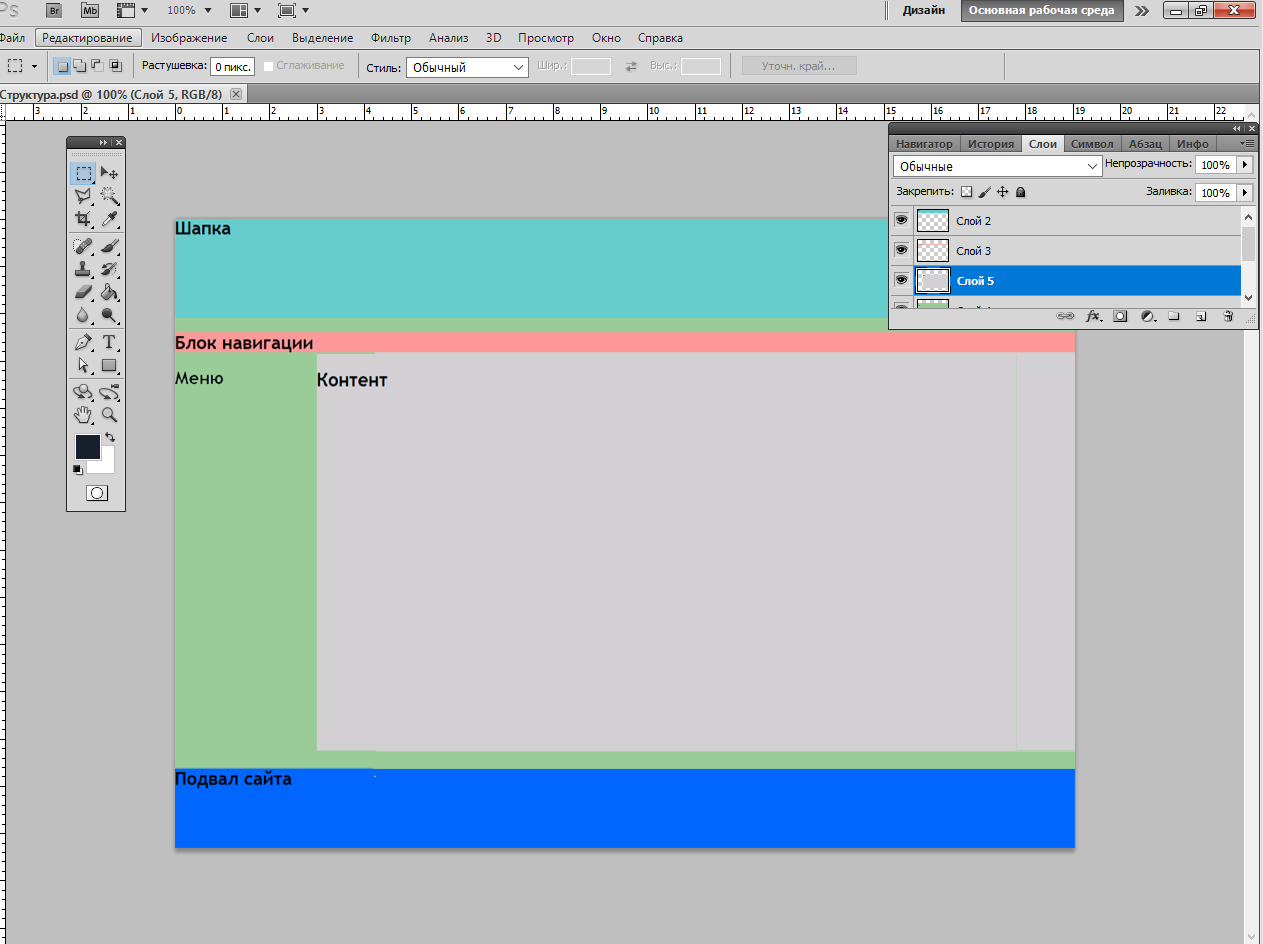
* Дизайнер надсилає макет сторінки, який є зображенням (часто розробленим в Photoshop або аналогічних засобах);
* Розробник реалізує зовнішній вигляд, описаний на макеті, зокрема з використанням HTML та CSS

Тому вміння реалізовувати з використанням HTML і CSS заданий дизайнером зовнішній вигляд сторінки є важливим для frontend розробників.

Варто враховувати, що макет зазвичай зображає загальний вигляд сторінки, а не точно задає як має виглядати остаточна сторінка. Зокрема, це стосується розмірів окремих елементів. Зазвичай макет описує відносну ширину окремих блоків, тоді як висота блоків визначається обсягом відповідного контенту (тексту, зображень тощо).

**Приклад верстки структури сайту за допомогою HTML і CSS.**

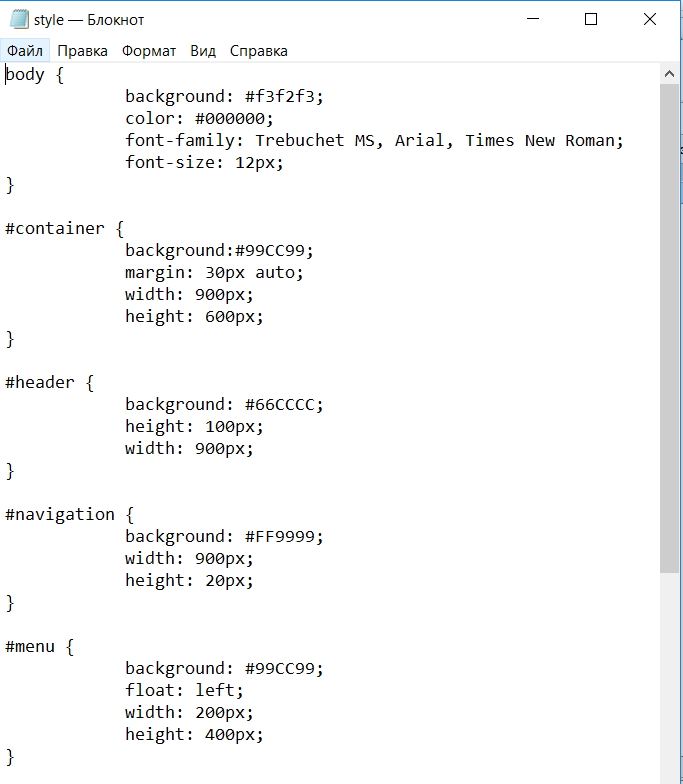
Макет в PS.

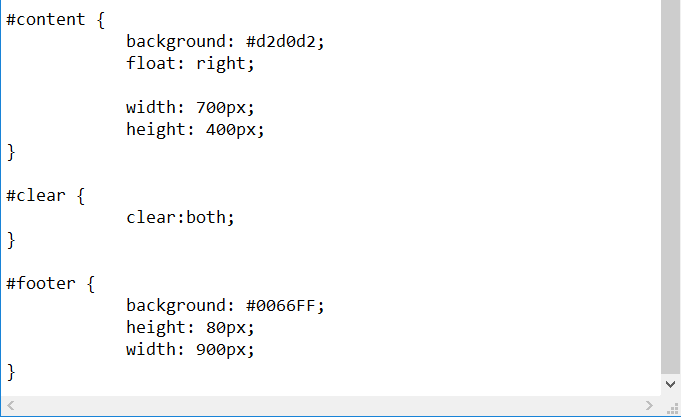


**Файл HTML**



Файл CSS.





**Оформлення звіту та порядок захисту**

**Оформлення звіту та порядок захисту** Лабораторна робота виконується на комп’ютері. Звіт оформлюється на аркушах А4 , в якому стисло відображається зміст, хід роботи та отримані результати. **Обов’язково навести посилання на проект в Git-hub та посилання на створений сайт.**

# **Лабораторна робота №4**

# **Тема: «Доступ до об'єктів і сценаріям. Доступ до властивостей і методів об'єктів»**

Мета роботи - навчитися використовувати різні способи доступу до властивостей і методів об'єктів для внесення змін в HTML-документ.

Завдання1.

У код лаб. роб. №1 додати два файли style.css та script.js.

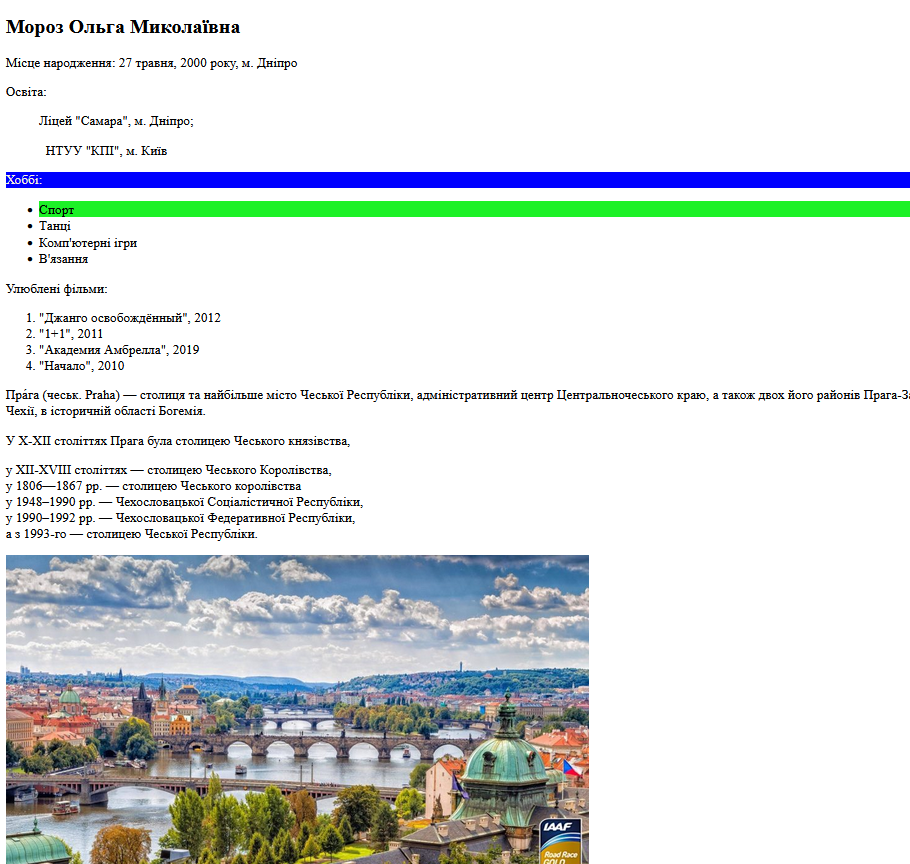
За допомогою JS реалізувати такі дії:

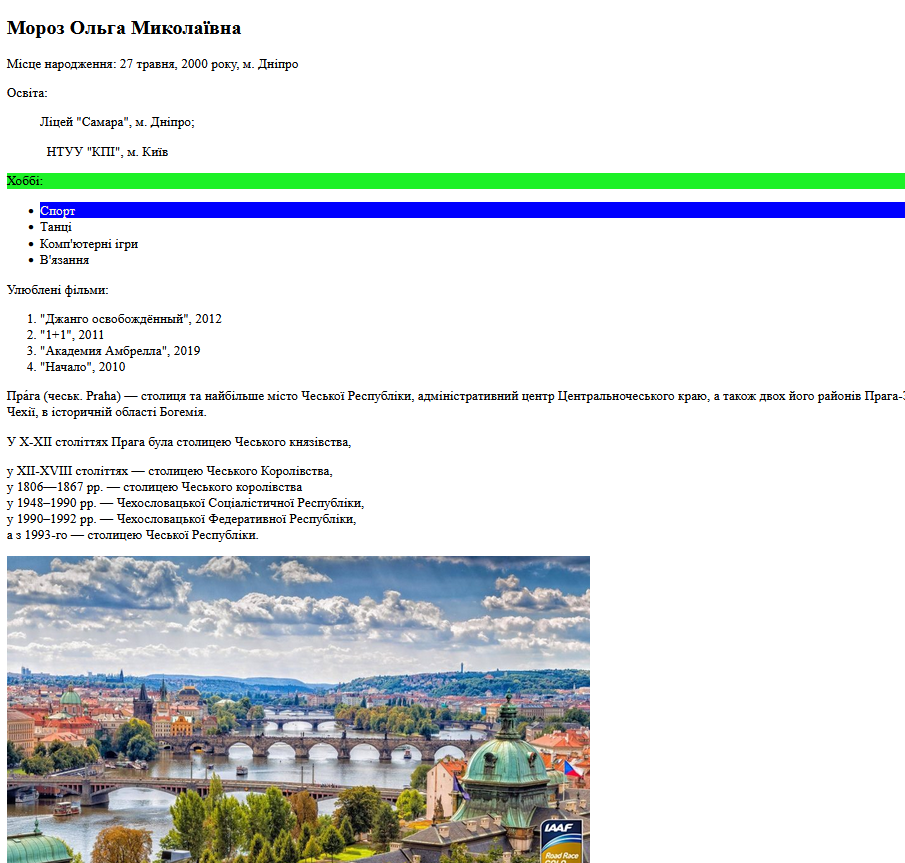
При першому кліку на елементі сторінці, що має номер по порядку рівний (n mod 10)+1, де n- номер варіанта змінити колір фону та тексту використовуючи метод getElementById (), а при кліку на наступному елементі змінити колір фону та тексту використовуючи метод метод querySelector(). При повторному кліку на відповідних елементах їх кольори фону та тексту змінюються.

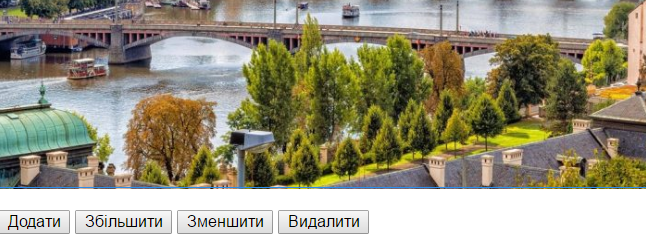
Завдання2.

Під зображенням додати 4 кнопки, які виконують наведені нижче дії: додати, збільшити, зменшити, видалити зображення.

Приклад виконання лабораторної роботи:







**Додаткова інформація**

**Пошук HTML-елемент за ідентифікатором**

Сам простий спосіб знайти HTML-елемент у DOM - це використовувати ідентифікатор елемента.

У цьому прикладі виконується пошук елементів із ідентифікатором = "Intro":

**Приклад**

var myElement = document.getElementById("intro");

<!DOCTYPE html>

<html>

<body>

<p id="intro">Hello World!</p>

<p>This example demonstrates the <b>getElementById</b> method!</p>

<p id="demo"></p>

<script>

var myElement = document.getElementById("intro");

document.getElementById("demo").innerHTML =

"The text from the intro paragraph is " + myElement.innerHTML;

</script>

</body>

</html>

**Пошук елементів HTML за допомогою селекторів CSS**

Якщо потрібно знайти всі елементи HTML, CSS, що відповідають указаному селектору (ідентифікатор, імена класів, типи, атрибути, значення атрибутів тощо), використовуйте метод querySelectorAll ().

У цьому прикладі повертається список усіх <p> елементів з класом = "intro".

**Приклад**

var x = document.querySelectorAll("p.intro");

<!DOCTYPE html>

<html>

<body>

<p>Hello World!</p>

<p class="intro">The DOM is very useful.</p>

<p class="intro">This example demonstrates the <b>querySelectorAll</b> method.</p>

<p id="demo"></p>

<script>

var x = document.querySelectorAll("p.intro");

document.getElementById("demo").innerHTML =

'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;

</script>

</body>

</html>

**Оформлення звіту та порядок захисту**

Лабораторна робота виконується на комп’ютері та на аркушах А4 оформлюється звіт, в якому стисло відображається зміст, хід роботи та отримані результати.

# **Лабораторна робота №5**

# **Тема: JavaScript. «Події. Регулярні вирази»**

Мета роботи - навчитися використовувати наявні в моделі документа події для внесення змін в сторінку.

**Завдання1.**

Вибрати рядок у таблиці 1, номер якого збігається з Вашим варіантом. Для формування форми взяти рядки, які відповідають Вашому варіанту таблиці2.

З отриманої інформації сформувати форму. За допомогою регулярних виразів JS виконати перевірку валідності введеної інформації. Якщо все правильно – вивести введену інформацію в окремому вікні, якщо є помилки – виділити рядки, які містять помилку.

Таблиця1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № вар. | Номера рядків в табл. 2 | | | | |
| 1 | 1 | 2 | 3 | 4 | 5 |
| 2 | 1 | 3 | 4 | 5 | 6 |
| 3 | 1 | 4 | 5 | 6 | 7 |
| 4 | 1 | 5 | 6 | 7 | 8 |
| 5 | 1 | 6 | 7 | 8 | 9 |
| 6 | 1 | 7 | 8 | 9 | 10 |
| 7 | 1 | 2 | 4 | 6 | 8 |
| 8 | 1 | 3 | 5 | 7 | 9 |
| 9 | 1 | 2 | 3 | 6 | 7 |
| 10 | 1 | 3 | 4 | 8 | 9 |
| 11 | 1 | 4 | 6 | 7 | 8 |
| 12 | 1 | 5 | 7 | 8 | 9 |
| 13 | 1 | 3 | 4 | 6 | 8 |
| 14 | 1 | 4 | 5 | 7 | 9 |
| 15 | 1 | 3 | 6 | 8 | 10 |

Таблиця2

|  |  |  |
| --- | --- | --- |
| 1 | ПІБ | ТТТТТТ Т.Т. |
| 2 | Варіант | ЧЧ |
| 3 | Група | ТТ-ЧЧ |
| 4 | Телефон | (ЧЧЧ)-ЧЧЧ-ЧЧ-ЧЧ |
| 5 | ID-card | ТТ №ЧЧЧЧЧЧ |
| 6 | Факультет | ТТТТ |
| 7 | Дата народж. | ЧЧ.ЧЧ.ЧЧЧЧ |
| 8 | Адреса | м. ЧЧЧЧЧЧ |
| 9 | e-mail | тттттт@ттттт.com |
| 10 | Telegram | @Т\_ТТТТТ |

Де Т-текст, Ч-число

**Завдання2.**

Створити таблицю розміром (6х6). Клітинки таблиці заповнюються послідовно номерами від 1 до 36 по рядках. При наведенні на клітинку, що відповідає номеру варіанта виконується зміна кольору на випадковий, при Click на ній - зміна кольору на обраний з палітри, а при dblClick:

Варіанти:

1. зміна кольору відповідного рядка таблиці;

2. зміна кольору відповідного стовпця таблиці;

3. зміна кольору клітинок головної діагоналі таблиці;

4. зміна кольору клітинок побічної діагоналі таблиці;

5. зміна кольору клітинок таблиці крім обраної;

6. зміна кольору всіх клітинок прямокутника, утвореного, починаючи з вибраної комірки таблиці;

7. зміна кольору рядків таблиці, починаючи з обраного, через один;

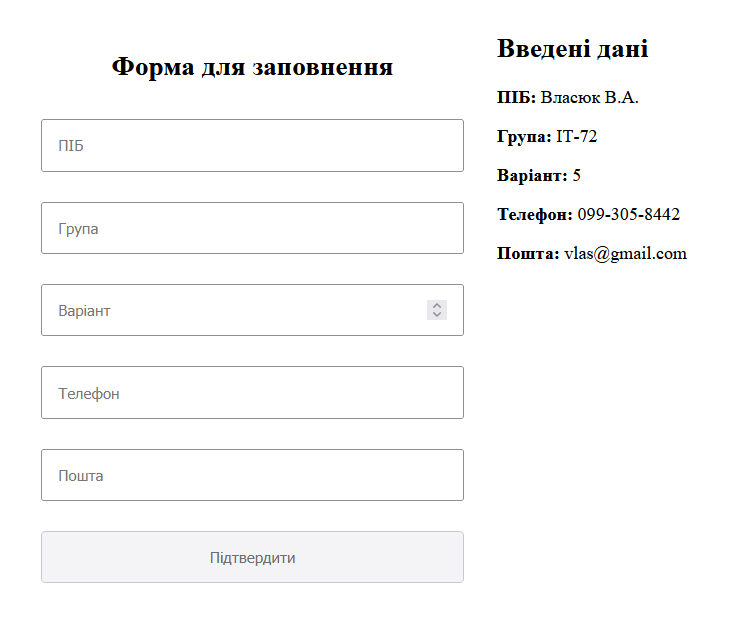
8. зміна кольору стовпців таблиці, починаючи з обраного, через один;

9. зміна кольору клітинок в обраному стовпці таблиці, починаючи з обраної, через одну;

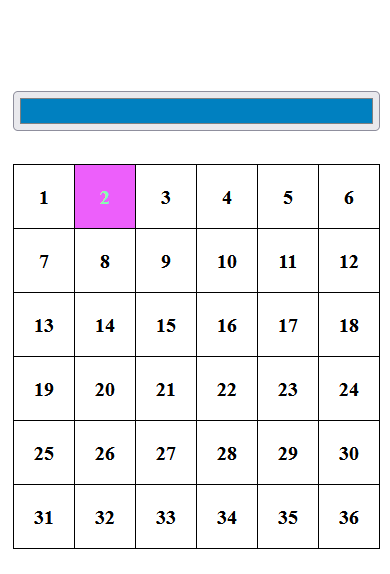
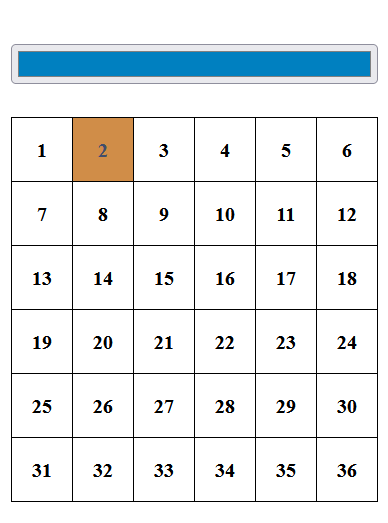
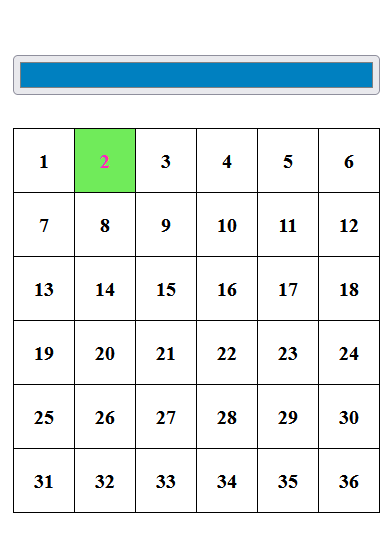
10. зміна кольору клітинок в вибраному рядку таблиці, починаючи з обраної, через одну.

**Приклад виконання лабораторної роботи:**

Перевірка даних на валідність.

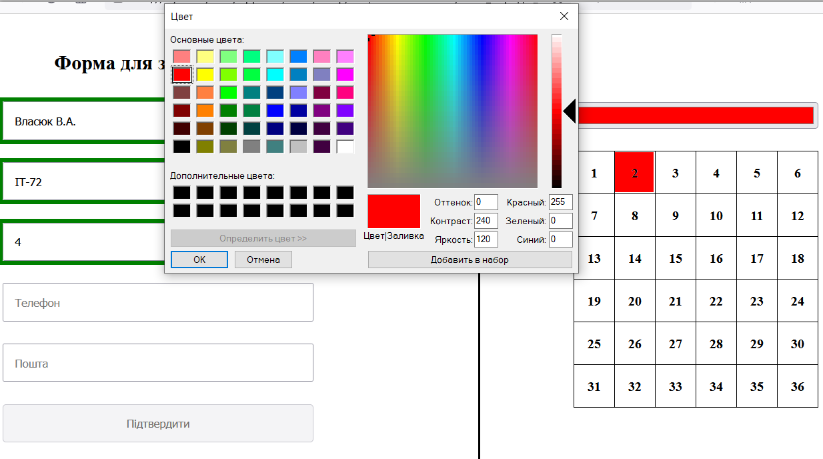


Зафарбування “рандомним” кольором (при наведенні)

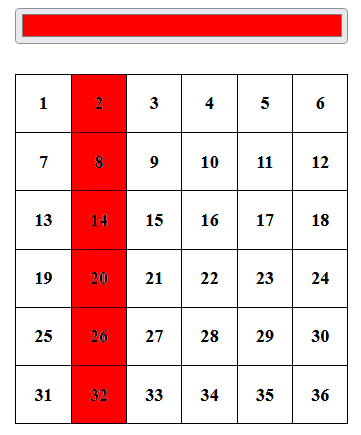
  

Зафарбування вибраним кольором (подія **Click**).

.



Зміна кольору рядка (для 2-го варіанту) (подія **dblClick** )



**Додаткова інформація**

**Реагування на події**

JavaScript може бути виконаний при виникненні події, наприклад, коли користувач клацає елемент HTML.

Щоб виконати код, коли користувач клацає на елемент, додайте код JavaScript в атрибут події HTML:

onclick=*JavaScript*

Приклади подій HTML:

* Коли користувач клацає мишею
* При завантаженні веб-сторінки
* Коли зображення було завантажене
* Коли миша переміщується над елементом
* При зміні поля введення
* При відправці HTML-форми
* Коли користувач натискає клавішу

У цьому прикладі вміст елемента <H1> змінюється, коли користувач клацає по ньому:

приклад

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 onclick="this.innerHTML = 'Ooops!'">Click on this text!</h1>  
  
</body>  
</html>

У цьому прикладі функція викликається з обробника подій:

приклад

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 onclick="changeText(this)">Click on this text!</h1>  
  
<script>  
function changeText(id) {  
    id.innerHTML = "Ooops!";  
}  
</script>  
  
</body>  
</html>

**Призначення подій за допомогою HTML DOM**

HTML DOM дозволяє призначати події елементам HTML за допомогою JavaScript:

приклад

Призначте подію OnClick елементу Button:

<script>  
document.getElementById("myBtn").onclick = displayDate;  
</script>

**Події OnLoad і Unload**

Події OnLoad і Unload ініціюються, коли користувач вводить або залишає сторінку.

Подія OnLoad може використовуватися для перевірки типу браузера відвідувача і версії браузера і завантаження відповідної версії веб-сторінки на основі інформації.

Події OnLoad і Unload можуть використовуватися для боротьби з файлами cookie.

приклад

<body onload="checkCookies()">

**подія onChange**

Подія onChange часто використовується в поєднанні з перевіркою полів введення.

Нижче наведено приклад використання OnChange. Функція верхнього регістру () буде викликатися, коли користувач змінює вміст поля введення.

приклад

<input type="text" id="fname" onchange="upperCase()">

**Події наведенні курсору миші**

<!DOCTYPE html>

<html>

<body>

<div onmouseover="mOver(this)" onmouseout="mOut(this)"

style="background-color:#D94A38;width:120px;height:20px;padding:40px;">

Mouse Over Me</div>

<script>

function mOver(obj) {

obj.innerHTML = "Thank You"

}

function mOut(obj) {

obj.innerHTML = "Mouse Over Me"

}

</script>

</body>

</html>

**Події OnMouseDown, OnMouseUp і OnClick**

Події OnMouseDown, OnMouseUp і OnClick є частинами клацання мишею. По-перше, при натисканні кнопки миші, ініціюється подія OnMouseDown, потім, коли кнопка миші звільняється, ініціюється подія OnMouseUp, і, нарешті, коли клацання миші завершується, ініціюється подія OnClick.

<!DOCTYPE html>

<html>

<body>

<div onmousedown="mDown(this)" onmouseup="mUp(this)"

style="background-color:#D94A38;width:90px;height:20px;padding:40px;">

Click Me</div>

<script>

function mDown(obj) {

obj.style.backgroundColor = "#1ec5e5";

obj.innerHTML = "Release Me";

}

function mUp(obj) {

obj.style.backgroundColor="#D94A38";

obj.innerHTML="Thank You";

}

</script>

</body>

</html>

Валідація форми.

Створюємо файл index.html і пишемо туди таку форму:

<form action='index.html' method="post">

1. Ваше ім’я:<br>

<input type="text" name="fnm" value="">

<br>

2. Повідомлення:<br>

<textarea name="text"></textarea>

<br>

3. E-Mail<br>

<input type="text" name="email" value="">

<br>

4. Контактний телефон<br>

<input type="text" name="phone" value="">

<br>

3. Введіть суму 10+10<br>

<input type="text" name="summa" id='summa' value=""><br>

<input type="hidden" name="hidden" value="ok">

<input type="submit" value="Надіслати" name="Submit" onClick="return Formdata(this.form)">

</form>

Створюємо файл data.js , пишемо в нього код перевірки:

function Formdata(data){

/\* якщо не заповнено поле Ваше им’я, довжина менше 3-x\*/

if (data.fnm != null && data.fnm.value.length < 3 )

{

alert('Заповніть поле "Ваше ім’я"');

return false;}

/\* якщо не заповнено поле Повідомлення \*/

if (data.text != null && data.text.value.length < 3)

{

alert('Заповніть поле "Повідомлення"');

return false;}

/\* e-mail Юзера \*/

if(data.email != null && data.email.value.length == 0)

{

alert('поле "E-Mail" пусте');

return false;}

if(data.email != null && data.email.value.length < 6)

{

alert('дуже малий "E-Mail"');

return false;}

if(!(/^w+[-\_.]\*w+@w+-?w+.[a-z]{2,4}$/.test(data.email.value)) )

{

alert("Введіть правильний E-Mail");

return false;}

/\* контактний телефон \*/

if(data.phone != null && data.phone.value.length == 0)

{

alert('поле "Контактний телефон" пусте');

return false;}

if(data.phone != null && data.phone.value.length < 5)

{

alert('поле "Контактний телефон" повинно містити мінімум п’ять символів');

return false;}

if(!(/^[0-9-+()s]+z/.test(data.phone.value+"z")))

{

alert('"Контактний телефон" вказано невірно');

return false;}

/\* робимо щоб поле сума дорівнювало числу \*/

number = document.getElementById("suma");

if (number.value !== '20')

{

alert('"Сума" не введена або введена невірно');

return false;}

}

**Оформлення звіту та порядок захисту** Лабораторна робота виконується на комп’ютері. Звіт оформлюється на аркушах А4 , в якому стисло відображається зміст, хід роботи та отримані результати. **Обов’язково навести посилання на проект в Git-hub та посилання на створений сайт.**

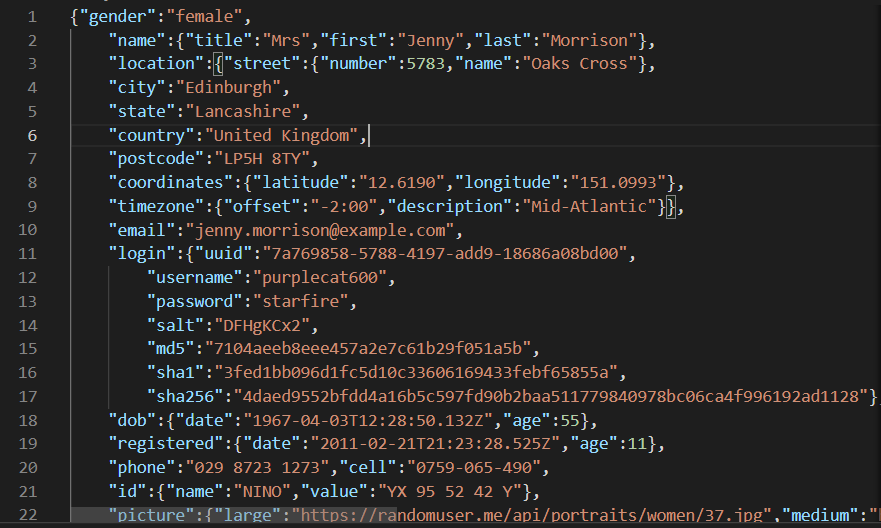
# **Лабораторна робота №6**

# **Тема: JavaScript. Формат JSON. API. Отримання даних із зовнішньої сторінки (fetch, promise)**

Мета роботи: навчитися працювати з даними в форматі **JSON**, отримувати та обробляти дані із зовнішньої сторінки за допомогою методів **fetch** та **promise**.

Завдання 1.

Створити js-скрипт для отримання інформації із зовнішньої сторінки <https://randomuser.me/api> за допомогою методів **fetch** та **promise** у форматі JSON.



Розробити виведення інформації на сторінку у відповідності до свого варіанту.

Табл.1

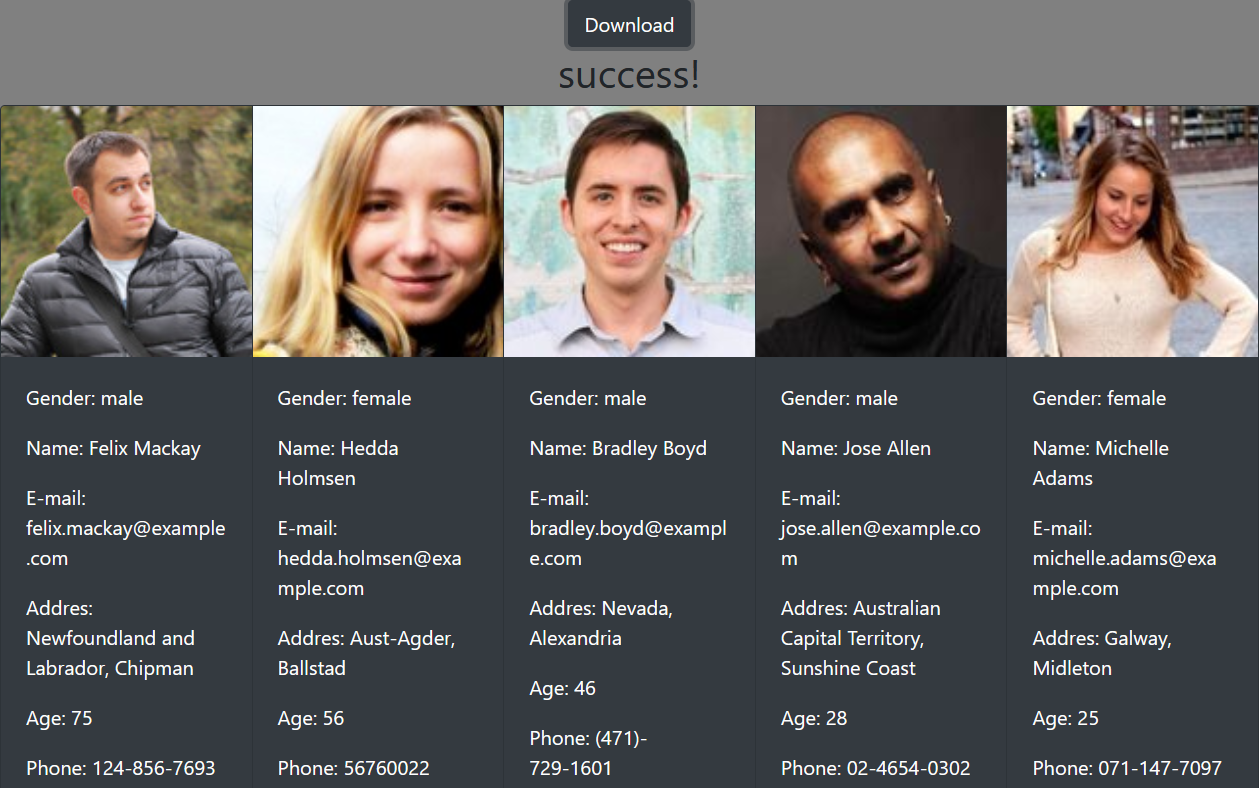
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ вар.** | **№ рядків в табл. 2** | | | | |
| 1 | 1 | 2 | 3 | 4 | 5 |
| 2 | 1 | 3 | 4 | 5 | 6 |
| 3 | 1 | 4 | 5 | 6 | 7 |
| 4 | 1 | 2 | 4 | 6 | 8 |
| 5 | 1 | 3 | 5 | 7 | 9 |
| 6 | 1 | 2 | 4 | 5 | 6 |
| 7 | 1 | 3 | 4 | 6 | 7 |
| 8 | 1 | 2 | 5 | 6 | 8 |
| 9 | 1 | 4 | 6 | 3 | 2 |
| 10 | 1 | 3 | 4 | 7 | 9 |
| 11 | 1 | 4 | 5 | 2 | 6 |
| 12 | 1 | 5 | 7 | 8 | 9 |
| 13 | 1 | 8 | 9 | 6 | 5 |
| 14 | 1 | 6 | 4 | 2 | 7 |
| 15 | 1 | 3 | 5 | 6 | 6 |
| 16 | 1 | 7 | 2 | 8 | 4 |
| 17 | 1 | 4 | 3 | 5 | 7 |
| 18 | 1 | 5 | 6 | 8 | 2 |
| 19 | 1 | 4 | 2 | 3 | 8 |
| 20 | 1 | 6 | 9 | 7 | 4 |

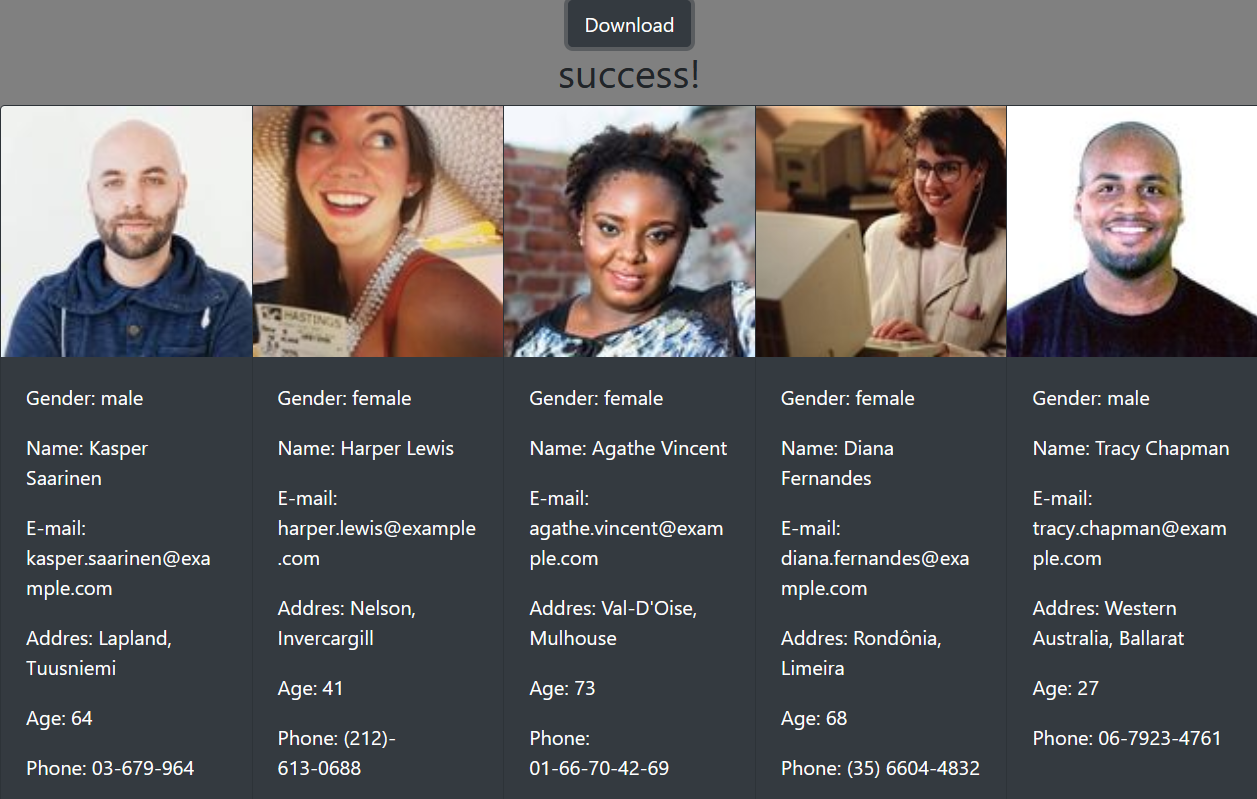
Табл.2

|  |  |
| --- | --- |
| 1 | **picture** |
| 2 | **name** |
| 3 | **cell** |
| 4 | **city** |
| 5 | **country** |
| 6 | **postcode** |
| 7 | **email** |
| 8 | **phone** |
| 9 | **coordinates** |

Алгоритм роботи: при натисканні на кнопку інформація зчитується з зовнішнього файлу у форматі JSON і записується на вашу сторінку згідно з Вашим варіантом.

Приклад виконання лаб. роб.





Теоретичні відомості

JSON існує як текстовий рядок (string), що необхідно при передачі даних по мережі. Він повинен бути перетворений у власний об’єкт JavaScript, якщо ви хочете отримати доступ до даних. Це не велика проблема. JavaScript надає глобальний об’єкт [JSON](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON) , який має методи перетворення між ними.

Об’єкт JSON може зберігатися у власному файлі, який в основному представляє собою текстовий файл із розширенням .jsonі [типом MIME](https://developer.mozilla.org/ru/docs/Glossary/MIME_type) application/json .

[Структура JSON](https://developer.mozilla.org/ru/docs/Learn/JavaScript/Objects/JSON#%25D1%2581%25D1%2582%25D1%2580%25D1%2583%25D0%25BA%25D1%2582%25D1%2583%25D1%2580%25D0%25B0_json)

Як описано вище, JSON представляє собою строку, формат якої дуже схожий на буквальний формат об’єкта JavaScript. Ви можете включати одні й ті ж базові типи даних всередині JSON, так само як і в стандартному об’єкті JavaScript — строки, числа, маси, булеви та інші об’єктні літери. Це дозволяє побудувати іерархію даних, к прикладу, так:

[

{

"name": "Molecule Man",

"age": 29,

"secretIdentity": "Dan Jukes",

"powers": [

"Radiation resistance",

"Turning tiny",

"Radiation blast"

]

},

{

"name": "Madame Uppercut",

"age": 39,

"secretIdentity": "Jane Wilson",

"powers": [

"Million tonne punch",

"Damage resistance",

"Superhuman reflexes"

]

}

]

[Fetch API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API) надає інтерфейс JavaScript для роботи із запитами та відповідями HTTP. Він також надає глобальний метод [fetch()](https://developer.mozilla.org/en-US/docs/Web/API/fetch) , який дозволяє легко та логічно отримувати ресурси по мережі асинхронно.

Подібна функціональність раніше досягається за допомогою [XMLHttpRequest](https://developer.mozilla.org/ru/docs/Web/API/XMLHttpRequest). Fetch являє собою найкращу альтернативу, яка може бути легко використана іншими технологіями, такими як [Service Workers](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API). Fetch також забезпечує єдине логічне місце для визначення інших зв’язаних із HTTP-початком, такі як CORS та розширення для HTTP.

Базовий запит на отримання даних дійсно простий в настройці. Перегляньте наступний код:

fetch('http://example.com/movies.json')

.then((response) => {

return response.json();

})

.then((data) => {

console.log(data);

});

Тут ми забираємо файл JSON по мережі і виводимо його вміст у консоль. Спосіб простого саме Використання  **fetch()** полягає у виклику функції з цим одним аргументом - рядком, що містить шлях до ресурсу, який ви хочете отримати - яка повертає обіцянку, що містить відповідь (об'єкт  [Response](https://developer.mozilla.org/ru/docs/Web/API/Response)).

[Promise](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Promise)- це  [об'єкт](https://developer.mozilla.org/ru/docs/Glossary/Object) , що повертається  [функція](https://developer.mozilla.org/ru/docs/Glossary/Function) , яка ще не завершив свою роботу. Проміс буквально являє собою "обіцянку", зроблена функція, яка через об'єкт промис поверне результат.

Коли викликана функція [асинхронно](https://developer.mozilla.org/ru/docs/Glossary/Asynchronous) завершує робота, викликається функція об'єкт проміс, званий обробник рішення (або виконання, або завершення) щоб вихідний об'єкт виклик знав, що завдання виконано

Об'єкт  Promise використовується для відкладених та асинхронних обчислень.

const promise1 = new Promise((resolve, reject) => {

setTimeout(() => {

resolve('foo');

}, 300);

});

promise1.then((value) => {

console.log(value);

// expected output: "foo"

});

console.log(promise1);

// expected output: [object Promise]

**Оформлення звіту та порядок захисту** Лабораторна робота виконується на комп’ютері. Звіт оформлюється на аркушах А4 , в якому стисло відображається зміст, хід роботи та отримані результати. **Обов’язково навести посилання на проект в Git-hub та посилання на створений сайт.**

# **Лабораторна робота №7**

# **Тема: Робота з React. Створення та робота з компонентами. Робота з props.**

**Завдання1.**

Завдання лаб. роб. №4 реалізувати за допомогою фреймворку React. Використовувати такі компоненти Header, Content, Image.

Компоненти Header, Image реалізувати у вигляді function, а компонент

Content – у вигляді class.

**Завдання2.**

Створити галерею товарів (не менш 6) на основі компонента GoodsCard та використати props для передачі параметрів. В картці товару повинно бути фото товару, назва, ціна.

**Теоретичні відомості.**

Для встановлення фреймворку вам знадобляться NodeJS не нижче версії 8.10 та npm не нижче версії 5.6 на вашому комп'ютері. Щоб створити проект у командному рядку, виконайте такі команди:

npx create-react-app my-app

cd my-app

npm start

Після цього у вас з'явиться папка my-app, що містить фреймворк. Знайдіть у цій папці папку *src* - це буде ваша робоча папка, в якій ви вестимете розробку вашого проекту.

У папці *src* знайдіть файл *App.js*. Найближчі уроки цей файл буде вашим основним робочим файлом. У ньому ви писатимете код, спостерігаючи його результати у вікні браузера.

Відкрийте цей файл у редакторі та видаліть з нього все зайве, привівши його ось до такого вигляду:

import React from 'react';

function App() {

return <div>

text

</div>;

}

export default App;

Після того, як ваш проект встановлений, для наступного запуску (наприклад, після перезавантаження комп'ютера) вам достатньо буде перейти через термінал у папку my-app та виконати наступну команду:

npm start

У React кожен такий блок називається *компонентом* . Кожен компонент може містити в собі дрібніші компоненти, ті в свою чергу ще дрібніші і таке інше.

Кожному компоненту React відповідає ES6 модуль, розташований в папці src. Ім'я файлу з модулем пишеться з великої літери та має відповідати функції, яка розташована у коді цього модуля.

Наприклад, файл з назвою *App.js* повинен містити в собі функцію App:

import React from 'react';

function App() {

return <div>

text

</div>;

}

export default App;

**Основний компонент**

Один із компонентів повинен бути основним - тим, до якого додаються інші компоненти. У React за замовчуванням таким компонентом буде компонент App.

**Макет сайту**

У папці *my-app/public* у файлі index.html розміщено макет сайту. Ви можете розміщувати в ньому будь-який HTML-код - і ви побачите результат цього коду в браузері.

Крім того, в макеті сайту є спеціальний див з *id* рівним *root*, в який *монтується* основний компонент. Під монтуванням розуміється те, що в цей *div* буде виводиться результат роботи нашого компонента.

**Результат роботи компонента**

Див з результатом буде виведено те, що повертає через *return* функцію компонента. У наступному прикладі це буде див з текстом:

function App() {

return <div>

text

</div>;

}

Для прикладу створимо React-додаток, що має такі компоненти: 

1. Navbar
2. MainContent
3. Footer

Компоненти, що виводяться App, повинні бути описані в окремих файлах, кожен з них повинен виводити якісь JSX-елементи.

В якості основи для вирішення цього завдання використовується стандартний проект, який створюється засобами create-react-app(якщо ви не знаєте як такий проект створити - погляньте на [цей](https://habr.com/company/ruvds/blog/432636/) матеріал). Тут використовується стандартний index.html.  
  
Код файлу index.js:

**import** React **from** "react"

**import** ReactDOM **from** "react-dom"

**import** App **from** "./App"

ReactDOM.render(

 <**App** />,

 document.getElementById("root")

)

Ось код файлу App.js. Зверніть увагу, що для зберігання файлів компонентів ми будемо використовувати папку components.

**import** React **from** "react"

**import** Header **from** "./components/Header"

**import** MainContent **from** "./components/MainContent"

**import** Footer **from** "./components/Footer"

**function** **App**() {

**return** (

       <**div**>

           <**Header** />

           <**MainContent** />

           <**Footer** />

       </**div**>

   )

}

**export** **default** App

Код файлу Header.js:

**import** React **from** "react"

**function** **Header**() {

**return** (

       <**header**>This is the header</**header**>

   )

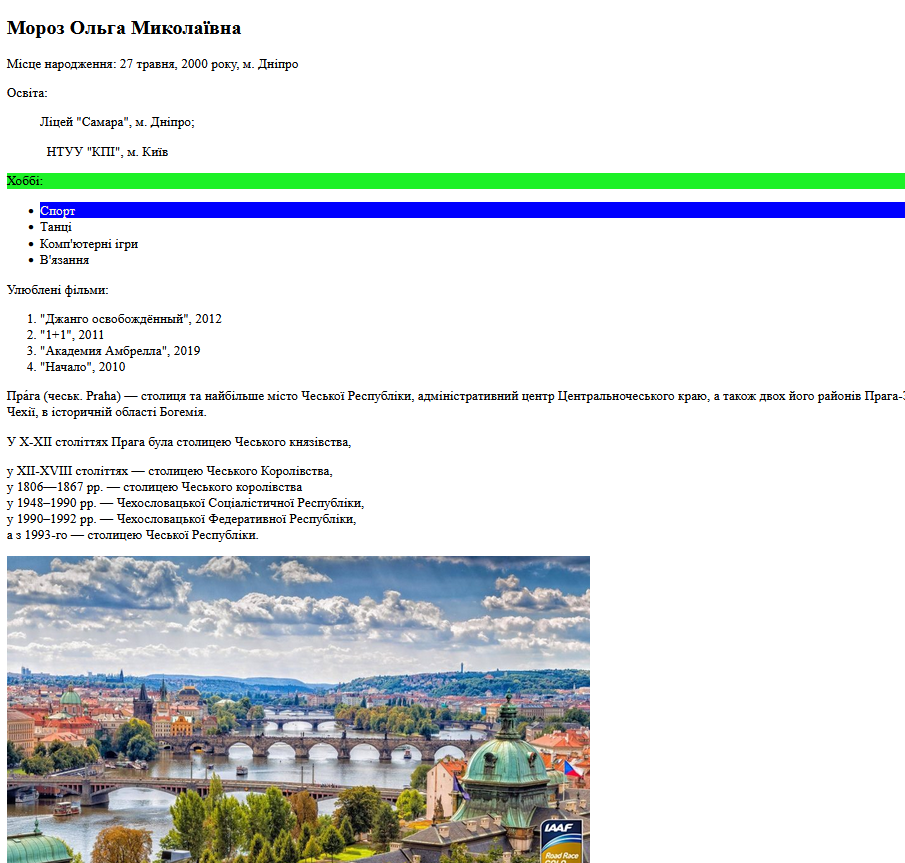
}

**export** **default** Header

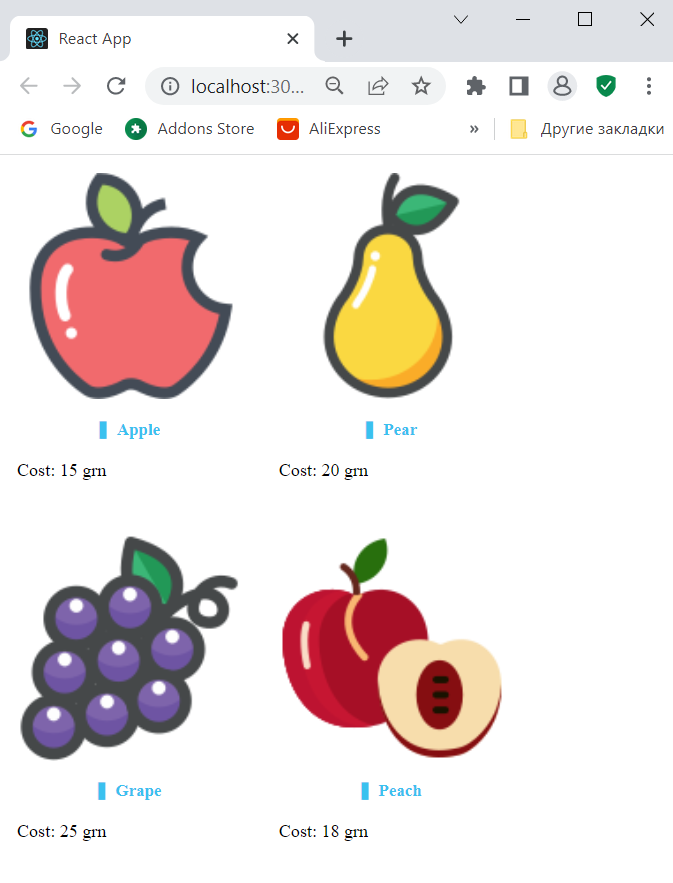
Аналогічно можна створити і інші компоненти ( MainContent,

Footer).

Приклад виконання завдання1.



Приклад виконання завдання2.



**Оформлення звіту та порядок захисту** Лабораторна робота виконується на комп’ютері. Звіт оформлюється на аркушах А4 , в якому стисло відображається зміст, хід роботи та отримані результати. **Обов’язково навести посилання на проект в Git-hub.**

# **Лабораторна робота №8**

# **Тема: Верстка сайту за допомогою HTML-5 і CSS-3 за зразком**

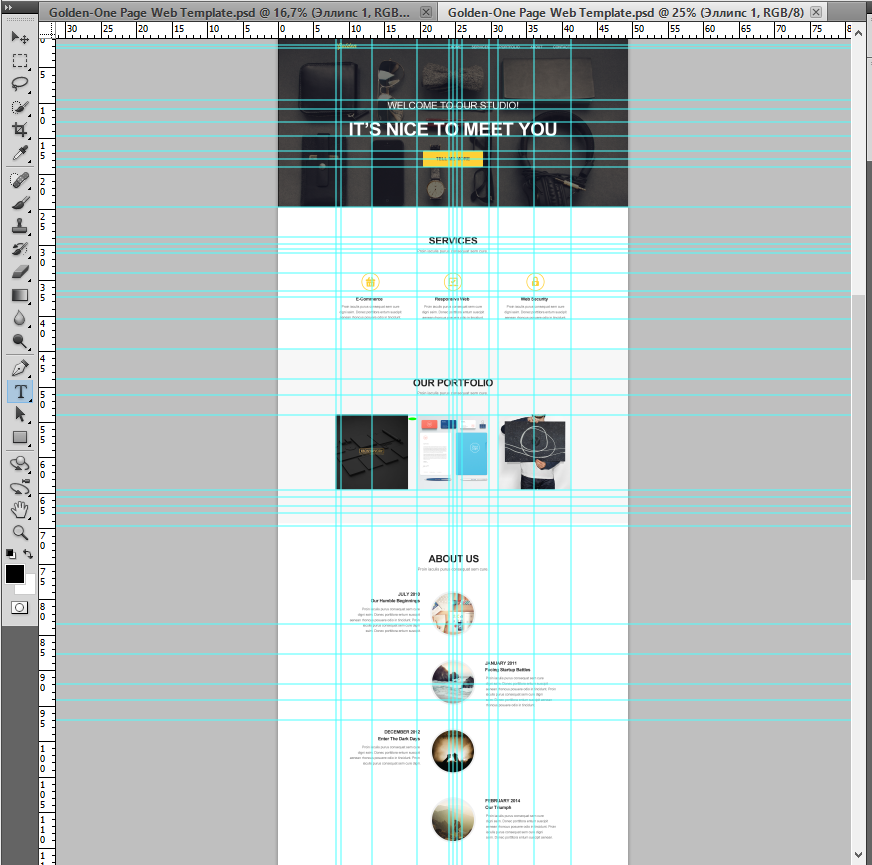
Завдання.

1. Зробити верстку сайту відповідно за своїм варіантом. (Зберегти дизайн сайту, текст та малюнки можуть відрізнятися від оригіналу).
2. \*Зробити адаптивний дизайн розробленого сайту.

|  |  |
| --- | --- |
| **Варіант** |  |
| **1,** | Рисунок 141 |
| **2** | Рисунок 139 |
| **3** | Рисунок 138 |
| **4** | Рисунок 137 |
| **5** | Рисунок 136 |
| **6** | Рисунок 135 |
| **7** | Рисунок 134 |
| **8** | Рисунок 133 |
| **9** | Рисунок 132 |
| **10** | Рисунок 131 |
| **11** | Рисунок 130 |
| **12** | Рисунок 129 |
| **13** | Рисунок 118 |
| **14** | Рисунок 116 |
| **15** | Рисунок 115 |
| **16** | Рисунок 104 |
| **17** | Рисунок 103 |
| **18** | Рисунок 102 |
| **19** | Рисунок 100 |
| **20** | Рисунок 156 |
| **21** | Рисунок 157 |
| **22** | Рисунок 158 |
| **23** | Рисунок 159 |
| **24** | Рисунок 167 |
| **25** | Рисунок 161 |
| **26** | Рисунок 162 |
| **27** | Рисунок 163 |
| **28** | Рисунок 164 |
| **29** | Рисунок 165 |
| **30** | Рисунок 166 |

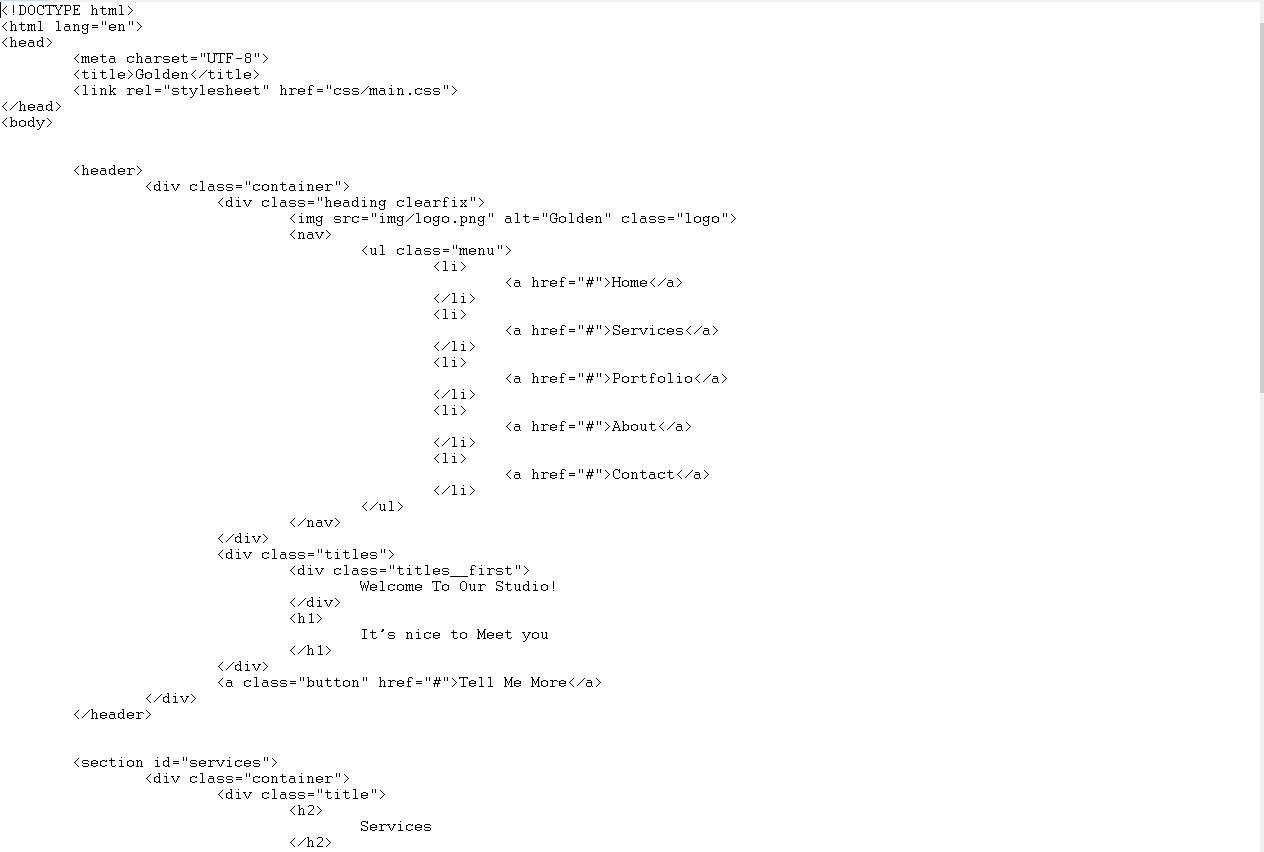
Приклад виконання лабораторної роботи.

Макет сайту в PS.

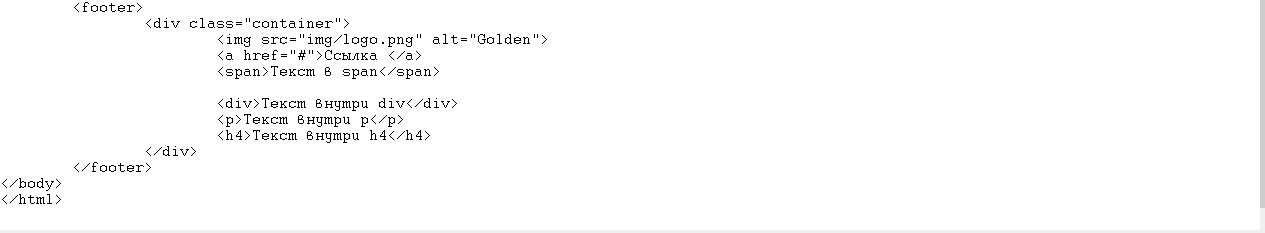


Приклад верстки сайту за допомогою HTML і CSS.

Файл HTML



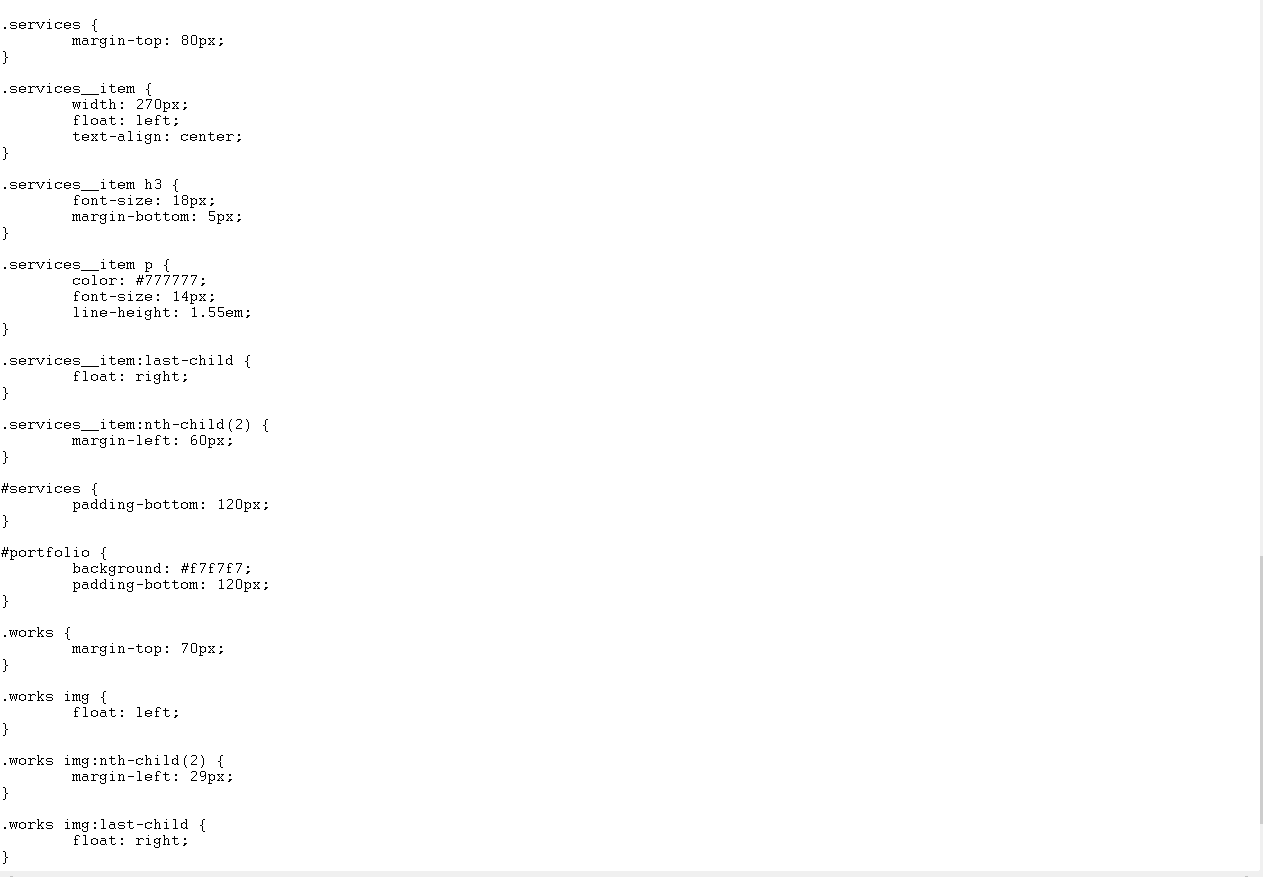




Файл CSS.









**Оформлення звіту та порядок захисту** Лабораторна робота виконується на комп’ютері. Звіт оформлюється на аркушах А4 , в якому стисло відображається зміст, хід роботи та отримані результати. **Обов’язково навести посилання на проект в Git-hub та посилання на створений сайт.**

# **Навчальний проєкт з курсу «Основи Frontend технологій»**

Необхідно реалізувати застосунок з використанням frontend технологій, а також можливо додаткових технологій (наприклад, взаємодія з backend, mobile або desktop клієнти, …). Тему проєкту, а також технології для реалізації можна обрати самостійно. Реалізований застосунок має використовувати адаптивний дизайн. Варто підготувати презентацію проєкту.

Навчальний проєкт варто виконувати в командах, орієнтовно до 3-4 учасників (але можна і самостійно). Якщо в команді більше учасників – то реалізований проєкт має бути відповідно складнішим, мати більше функціональних можливостей, використовувати більше технологій, або більш складні технології, або більш глибоко використовувати можливості певних технологій тощо.

В навчальному проєкті можна використовувати будь-які frontend технології, як ті що розглядались на лекціях, так і інші.