МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. СІКОСРЬКОГО»

Кафедра інформаційних систем та технологій

**МЕТОДИЧНІ ВКАЗІВКИ**

до виконання лабораторних робіт

з курсу **«****OсновиWEB-технологій/Основи front-end технологій»**

КИЇВ-2021

**Увага!** Варіанти лабораторних робіт відповідають **номеру студента за списком групи**. При написанні кода **html** у тегі **title** необхідно вказувати інформацію про студента (**ПІБ, група,номер лаб. роботи, номер варіанта**).

Звіти з лаб. роб. надсилати за адресою: **l.golubev@ukr.net**

**Лабораторна робота №1**

**з курсу «OсновиWEB/Основи front-end технологій»**

**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.

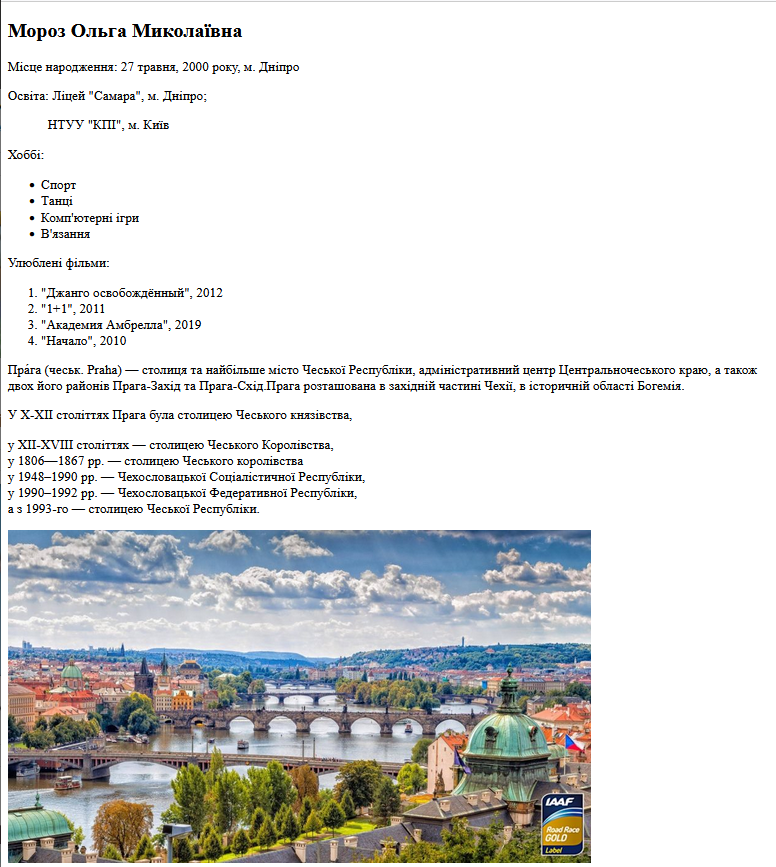
8. **Робота з Git.**

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.

**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**

**з курсу «OсновиWEB/Основи front-end технологій»**

Тема «Мова гіпертекстової розмітки сайту HTML-5. Таблиці.

Каскадна таблиця стилів 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/)

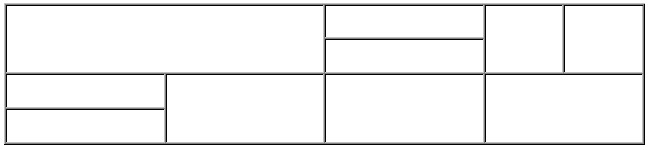
Зробити таким чином, щоб при наведенні курсора на 1-й та 2-й рядки кольори змінювалися на протилежні (використати псевдоклас hover).

**Завдвння2.**

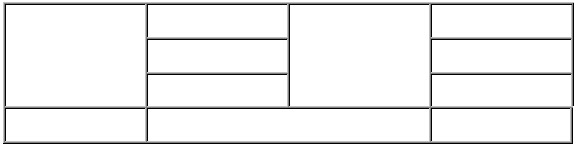
Створити зображення відповідно до свого варіанта: 1. – за допомогою position; 2- за допомогою transform , використовуючи тільки властивості CSS. Властивості CSS зберегти в окремому файлі style.css.

**Варіанти до завдання 1.**

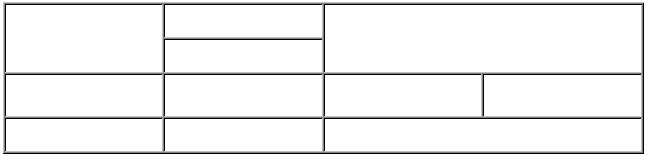
**Варіант 1**



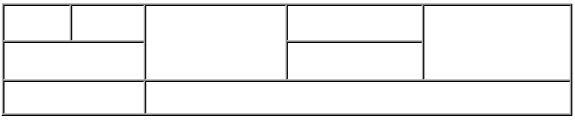
**Варіант 2**



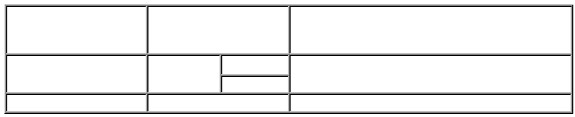
**Варіант 3**



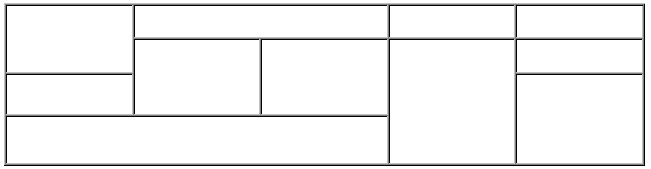
**Варіант 4**



**Варіант 5**



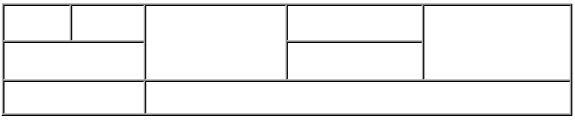
**Варіант 6**



**Варіант 7**



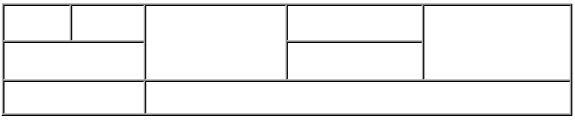
**Варіант 8**



**Варіант 9**



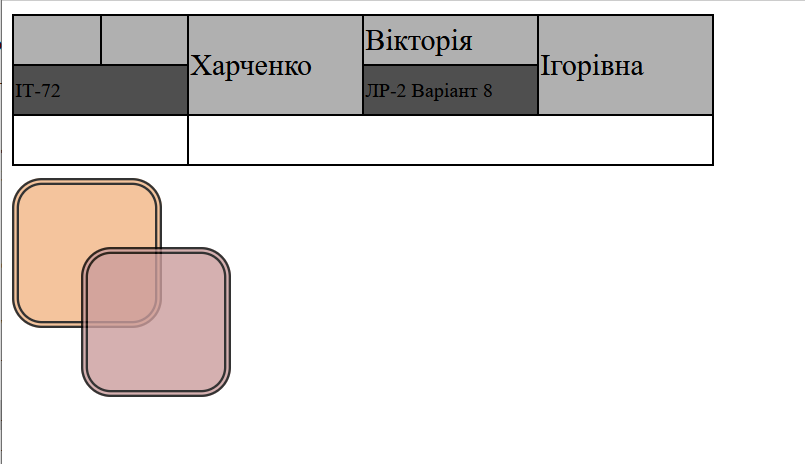
**Варіант 10**



**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>

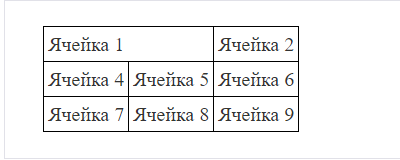
<td>Ячейка 7</td>

<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" \o "CSS властивість position) : 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**

**з курсу «OсновиWEB/Основи front-end технологій»**

**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 **сontainer** та **width** у відсотках.

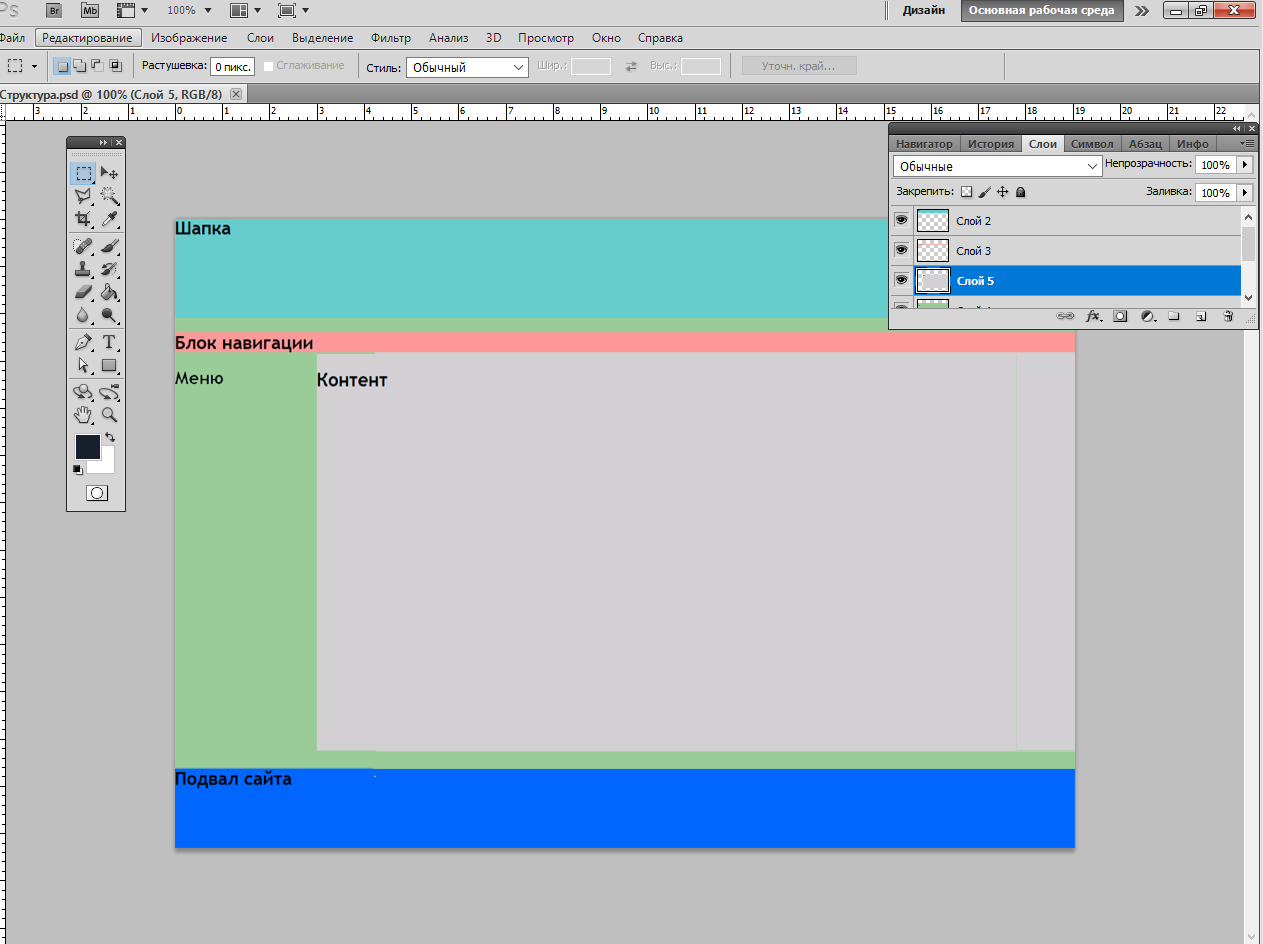
**Завдання 2.**

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

|  |  |
| --- | --- |
| Варіанти: |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7. |  |
| 8. |  |
| 9. |  |
| 10. |  |

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

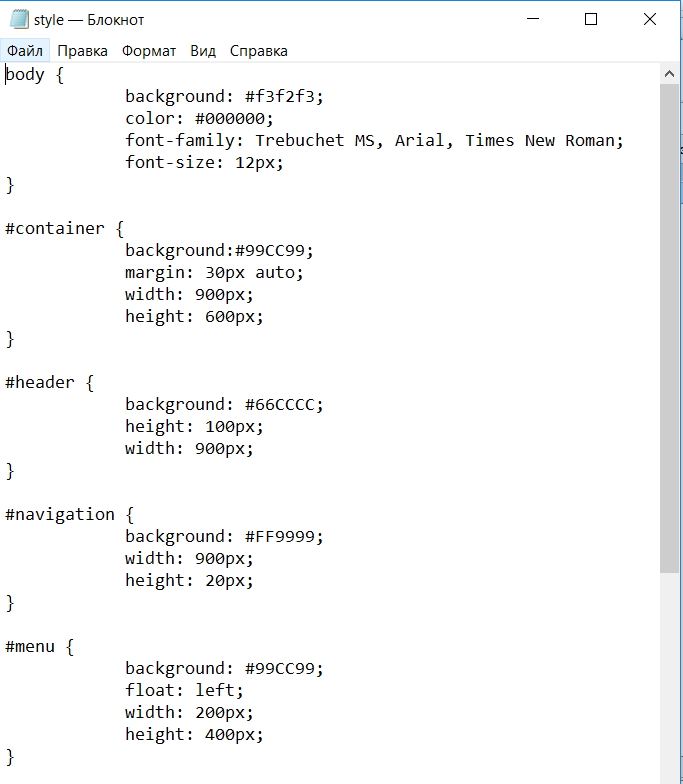
Макет в PS.

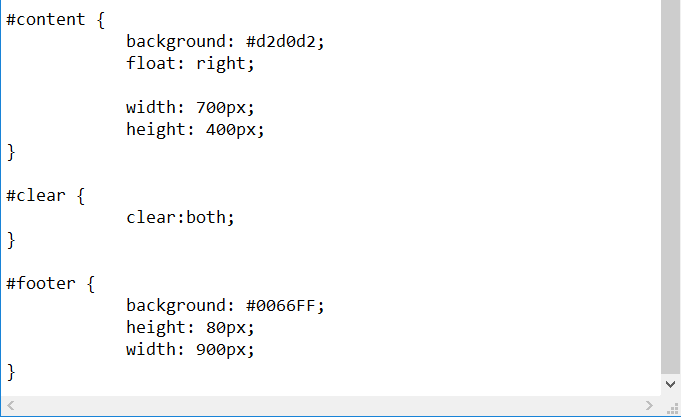


**Файл HTML**

re

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**

**з курсу «OсновиWEB/Основи front-end технологій»**

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

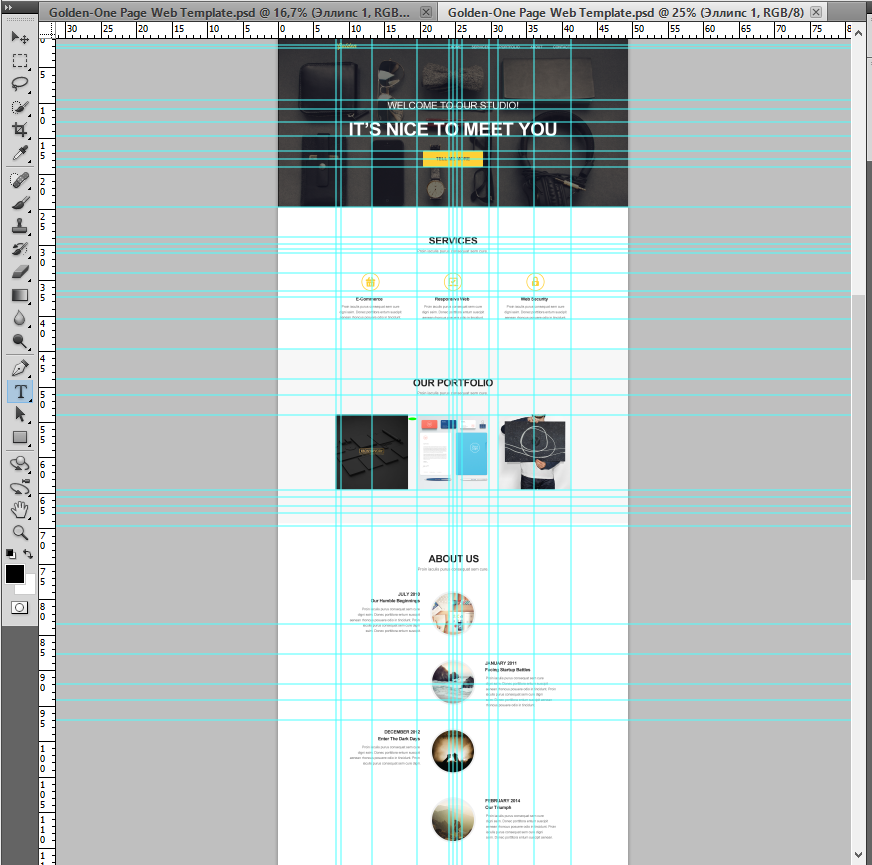
**Завдання.**

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,** |  |
| **2** |  |
| **3** |  |
| **4** |  |
| **5** |  |
| **6** |  |
| **7** |  |
| **8** |  |
| **9** |  |
| **10** |  |
| **11** |  |
| **12** |  |
| **13** |  |
| **14** |  |
| **15** |  |
| **16** |  |
| **17** |  |
| **18** |  |
| **19** |  |
| **20** |  |
| **21** |  |
| **22** |  |
| **23** |  |
| **24** |  |
| **25** |  |
| **26** |  |
| **27** |  |
| **28** |  |
| **29** |  |
| **30** |  |

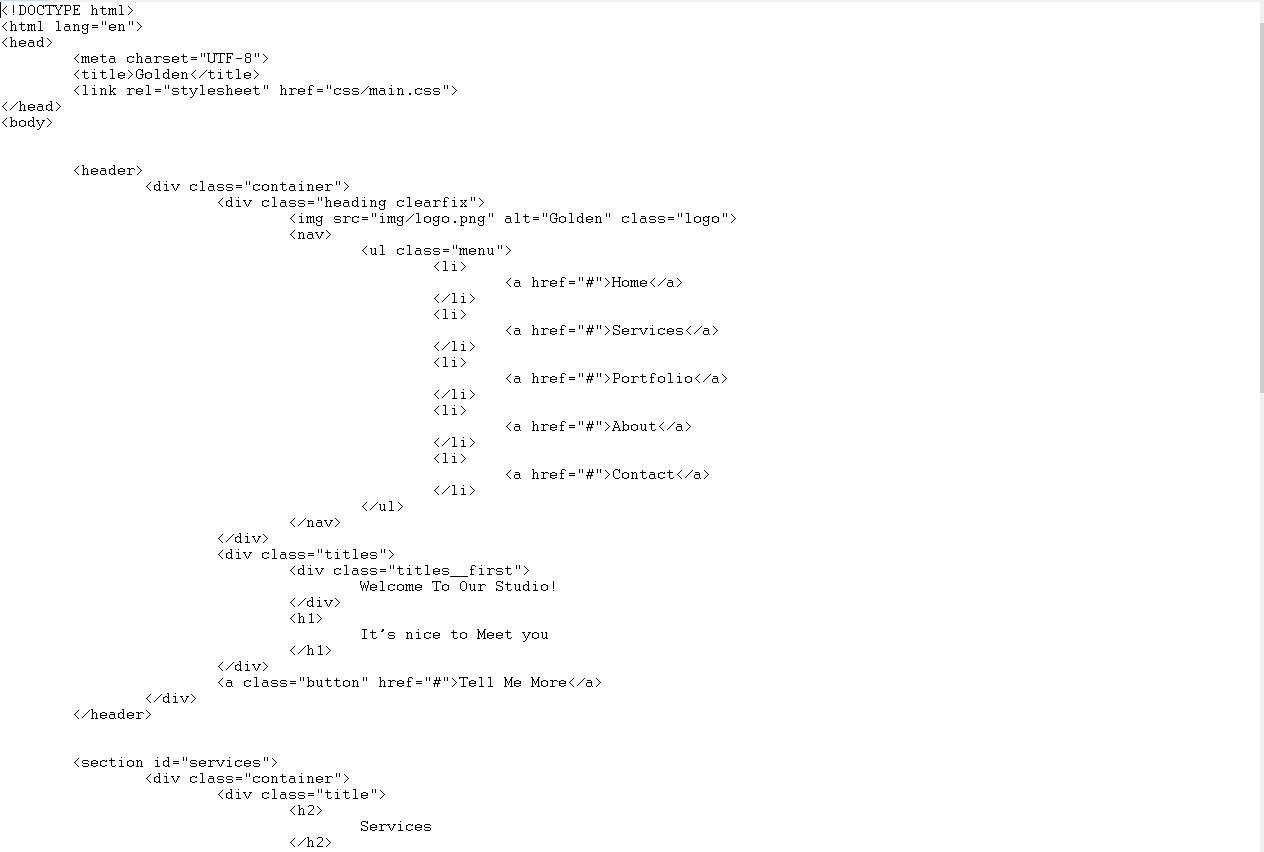
Приклад виконання лабораторної роботи.

Макет сайту в PS.

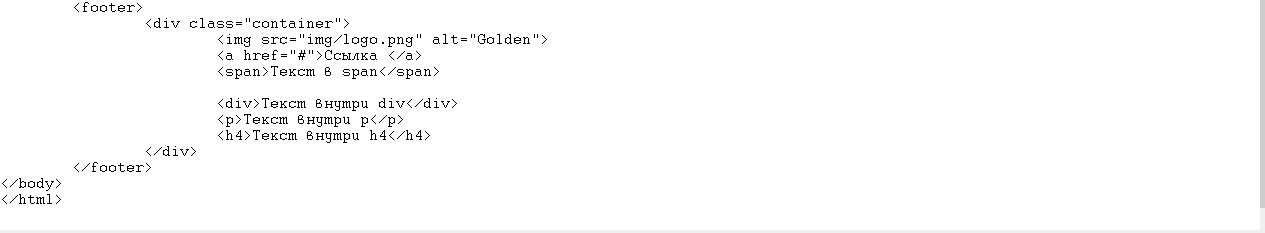


Приклад верстки сайту за допомогою HTML і CSS.

Файл HTML



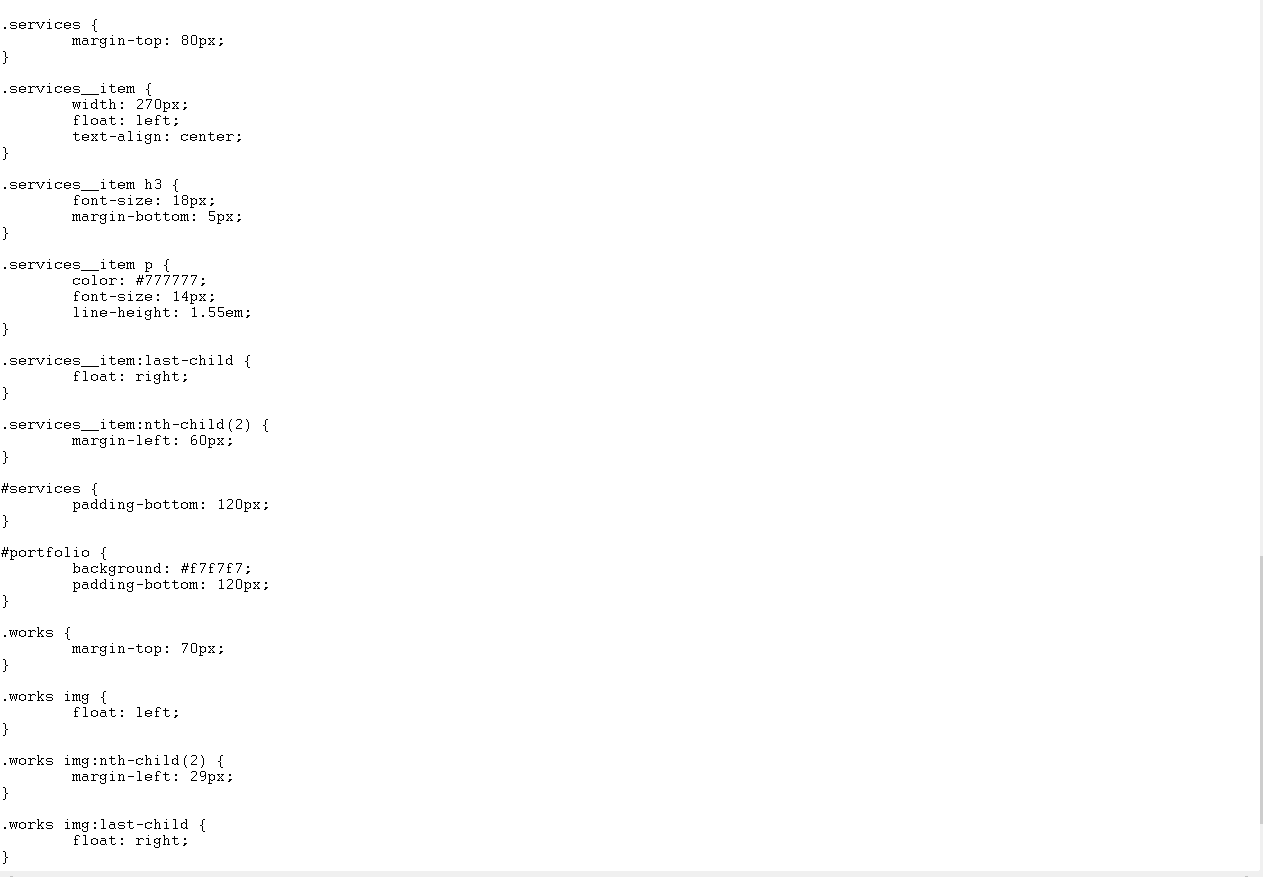




Файл CSS.









**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.**

**Лабораторна робота №5**

**з курсу «OсновиWEB/Основи front-end технологій»**

**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.**

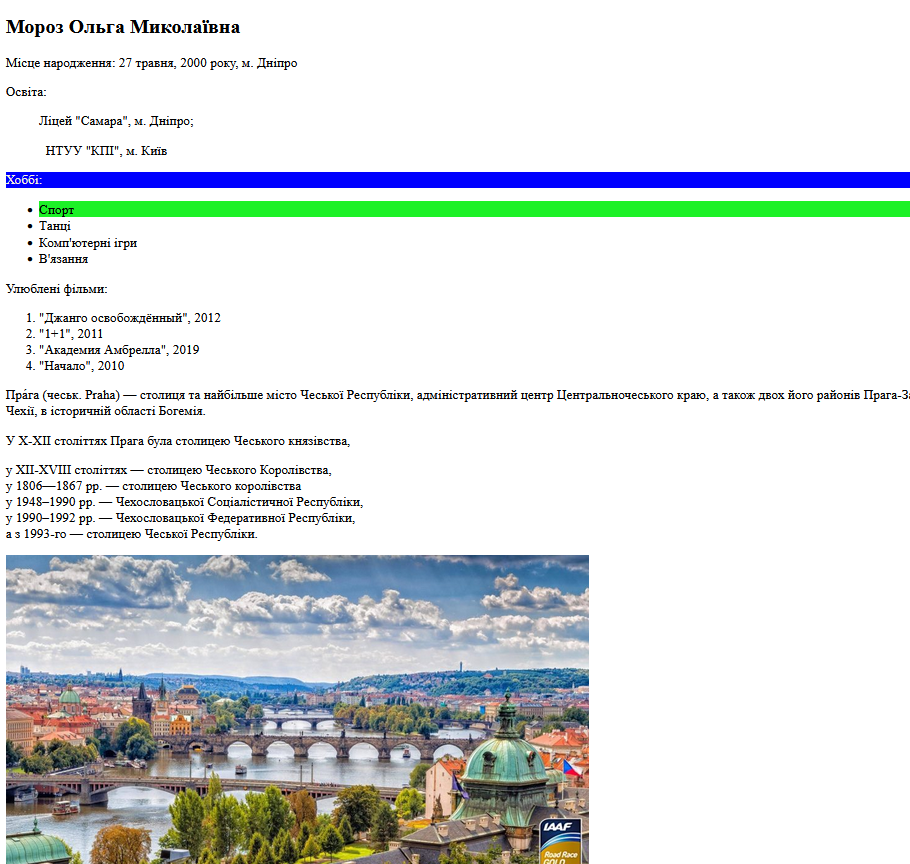
За допомогою **JS** реалізувати такі дії:

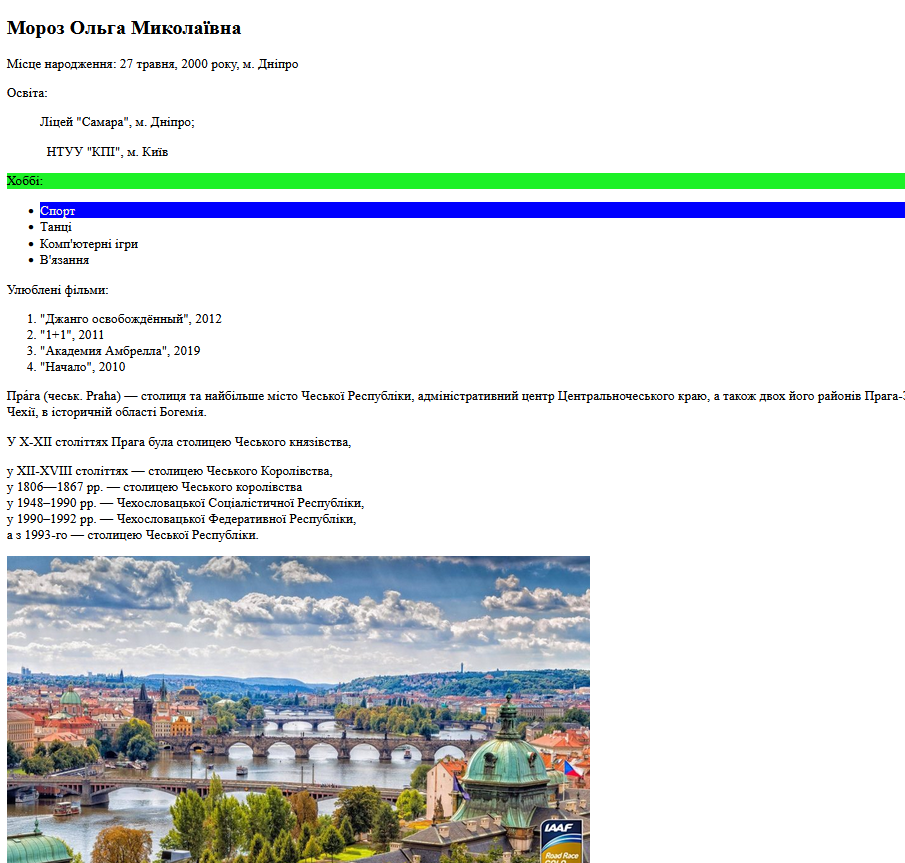
При першому кліку на елементі сторінці, що має номер по порядку рівний (n mod 10)+1, де n- номер варіанта змінити колір фону та тексту використовуючи метод getElementById (), а при кліку на наступному елементі змінити колір фону та тексту використовуючи метод метод querySelector(). При повторному кліку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

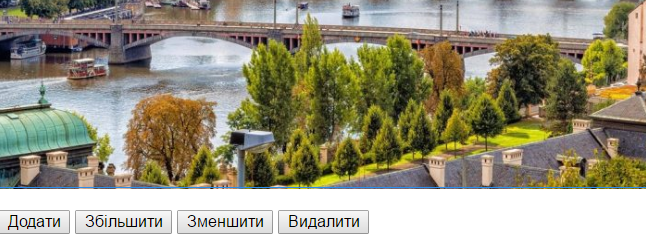
**Завдання2.**

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 №6**

**з курсу «OсновиWEB/Основи front-end технологій»**

**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;

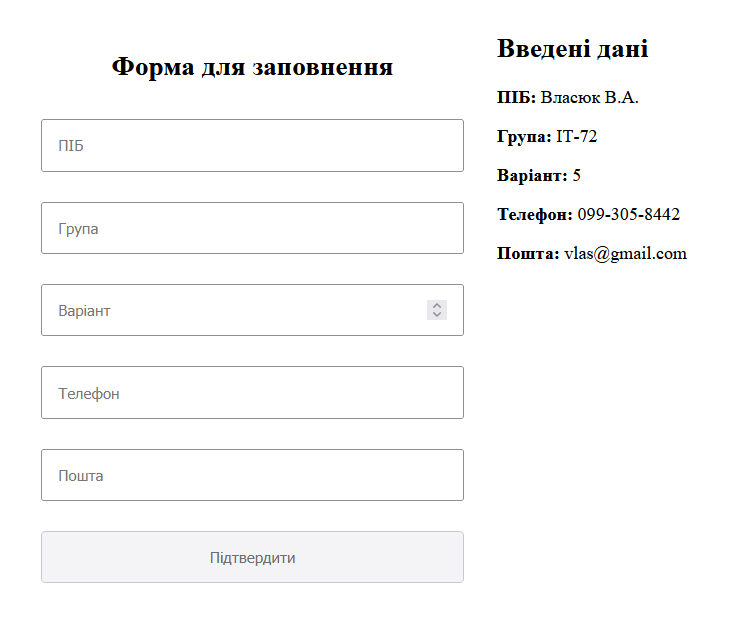
8. зміна кольору стовпців таблиці, починаючи з обраного, через один;

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

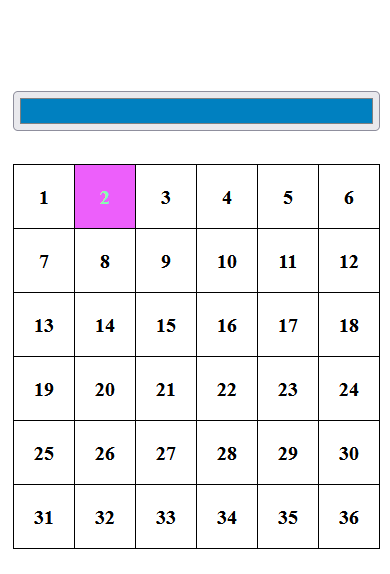
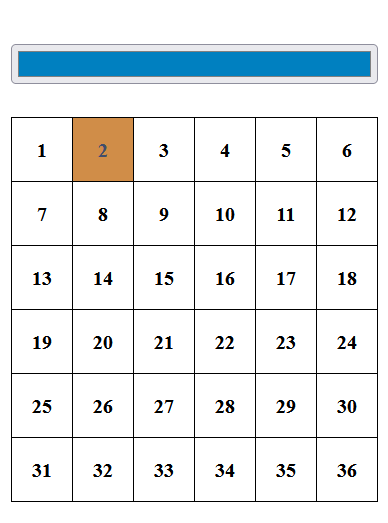
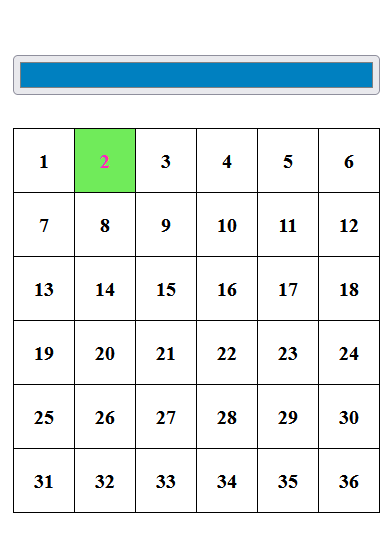
10. зміна кольору клітинок в вибраному рядку таблиці, починаючи з обраної, через одну.

**Приклад виконання лабораторної роботи:**

Перевірка даних на валідність.

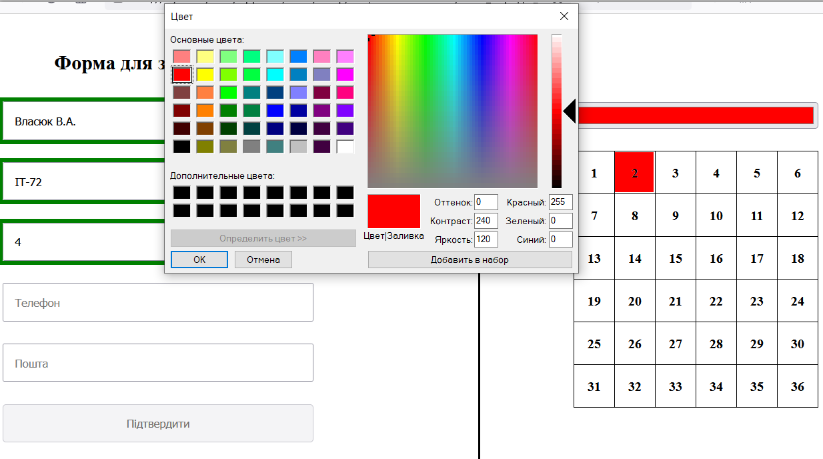


Зафарбування “рандомним” кольором (при наведенні)

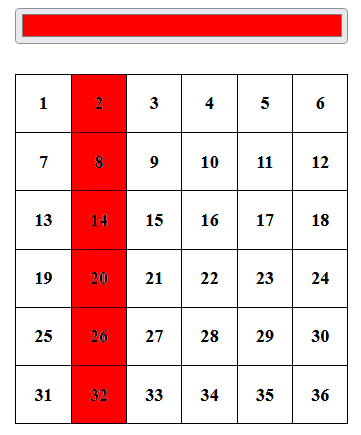
  

Зафарбування вибраним коліром (подія **Click**).

.



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



**Додаткова информація**

**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.php і пишемо туди таку форму:

<form action='index.php' 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="отправить" 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 им’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 №7**

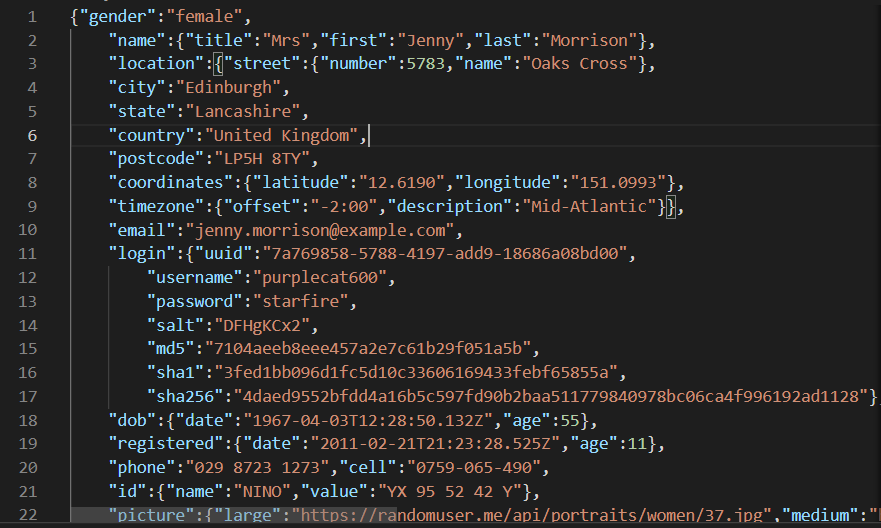
**з курсу «OсновиWEB/Основи front-end технологій»**

**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 **fitch** 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

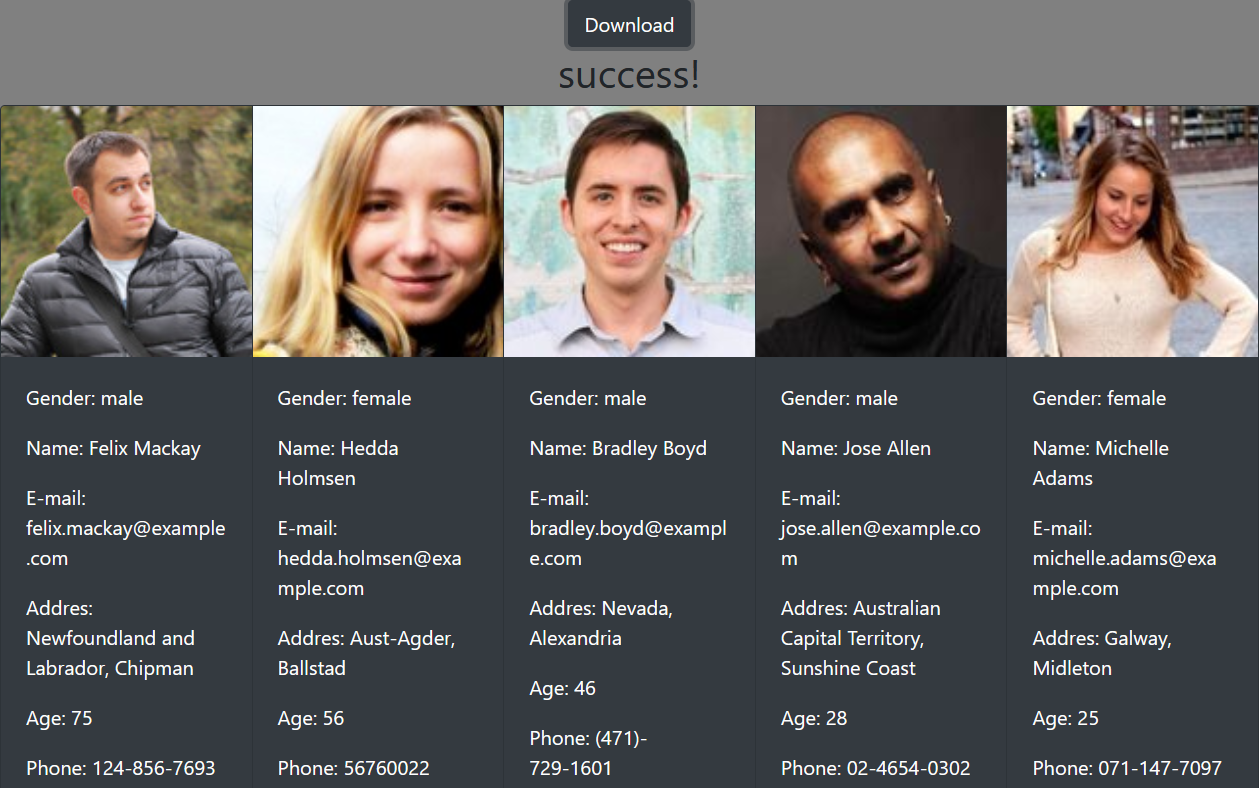
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ вар.** | **№ рядків в табл. 2** | | | | |
| 1 | 1 | 2 | 3 | 4 | 5 |
| 2 | 1 | 3 | 4 | 5 | 6 |
| 3 | 1 | 4 | 5 | 6 | 7 |
| 4 | 1 | 2 | 4 | 6 | 8 |
| 5 | 1 | 3 | 5 | 7 | 9 |
| 6 | 1 | 2 | 4 | 5 | 6 |
| 7 | 1 | 3 | 4 | 6 | 7 |
| 8 | 1 | 2 | 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 |

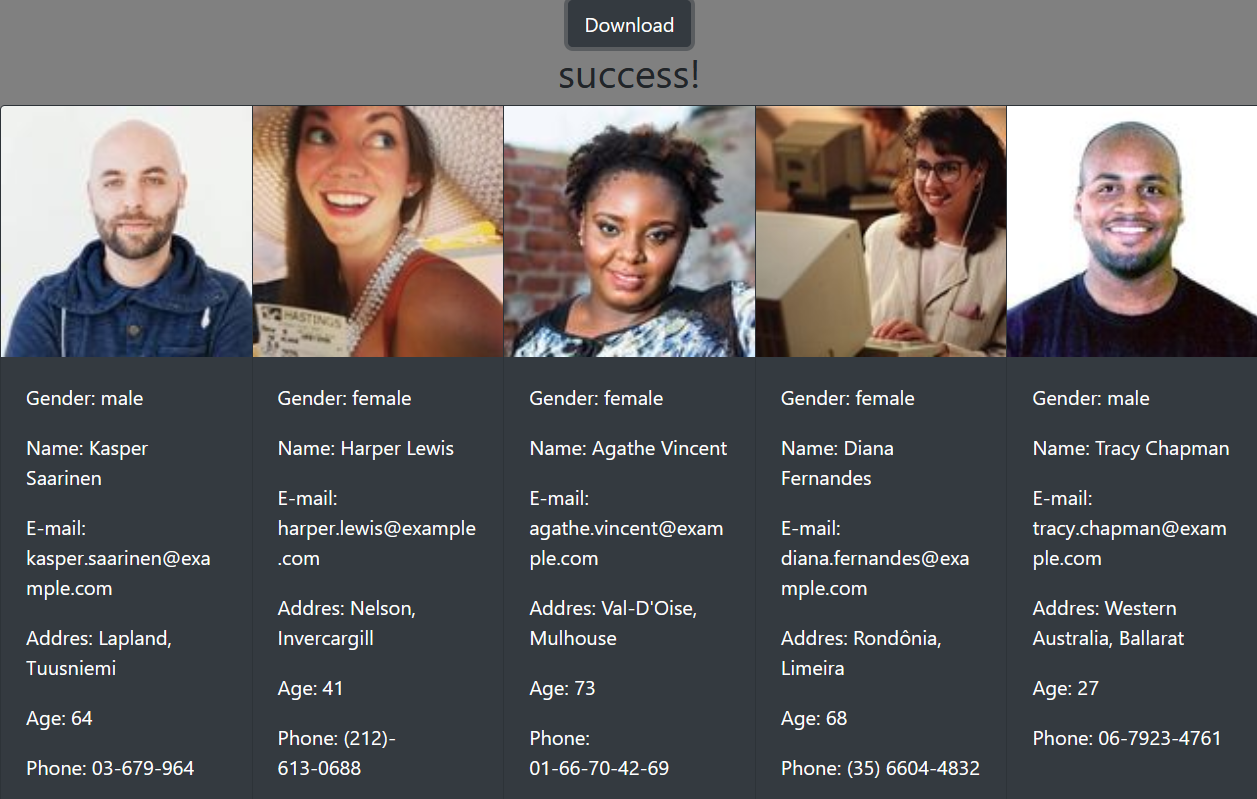
Табл.2

|  |  |
| --- | --- |
| 1 | **picture** |
| 2 | **name** |
| 3 | **location** |
| 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 строка, 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#%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_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() (en-US)](https://developer.mozilla.org/en-US/docs/Web/API/fetch" \o "Наразі доступно лише англійською (США)) , 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 (en-US)](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API" \o "Наразі доступно лише англійською (США)) . 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 - рядоком, 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.**

**Література**

1. Пол Вилтон, Джереми МакПик. JavaScript. Руководство программиста. СПб: Питер, 2009-720 с.

2. Стоян Стефанов. JavaScript. Шаблоны. СПб: Символ-плюс, 2011-272 с.

3. Дунаев В.. HTML, скрипты и стили. СПб: БХВ-Петербург, 2011- 816с.

4. Дронов В.. HTML 5, CSS 3 и Web 2.0. Разработка современных Webсайтов. СПб: БХВ-Петербург, 2011- 416 с.

5. Джон Поллок. JavaScript. Руководство разработчика. СПб: Питер, 2011-544 с.

6. Дэвид Макфарланд. JavaScript. Подробное руководство. Эксмо, 2009-608 с.

7. Климов А. JavaScript на примерах. СПб: БХВ-Петербург, 2009-336 с.

8. Шафер С., HTML, XHTMLи CSS. Библия пользователя. М.: Вильямс, 2010 – 656 с.

9. Лабберс К., Олберс Н., Салим К.. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений. М.: Вильямс, 2011 – 272 с.