

ЛАБОРАТОРНА РОБОТА №6

CSS позиціонування. CSS псевдокласи та псевдоеlementи

Теоретичний матеріал

Перелік основних CSS-властивостей:

1. opacity

Визначає рівень прозорості елемента веб-сторінки. При частковій або повній прозорості через елемент проступає фоновий малюнок або інші елементи, розташовані нижче напівпрозорого об'єкта.

Приймає значення від 0 до 1, де значення 0 – це повна прозорість елемента (він стає невидимим), 1 – повністю видимий елемент.

```
.example-block {  
  opacity: 0;  
  font-size: 30px;  
}
```

Ми побачимо, що наш блок «пропав», проте місце яке він займав, залишилось, іншими словами – блок не забирається із сторінки, а тільки стає прозорим, а займане ним місце залишається за блоком збереженим.

Це є основною відмінністю властивості opacity: 0; від властивості display: none; Властивість opacity застосовна до рядкових тегів.

2. visibility

Призначений для відображення або приховання елемента, включаючи рамку навколо нього і фон. При приховуванні елемента, хоча він і стає непомітний, місце, яке елемент займає, залишається за ним.

Може приймати значення:

- visible – блок видимий, встановлюється за замовчуванням.
- hidden – блок не видимий.
 - відмінність visibility: hidden; від display: none; в тому, що блок не забирається із сторінки, а тільки ховається, тобто займане ним місце залишається.
 - відмінність visibility: hidden; від opacity: 0; у тому, що блок ховається і взаємодіяти з ним не можна.

Властивість visibility застосовна до рядкових тегів.

3. position

Встановлює спосіб позиціонування елемента щодо вікна браузера чи інших об'єктів на веб-сторінці.

Властивість position використовують додатково із допоміжними властивостями top, bottom, left і right – які керують положенням елемента зверху знизу ліворуч і праворуч, іншими словами, дозволяють нам зсувати наш елемент як по-горизонталі так і по-вертикалі відносно його початкового положення.

Значення задають пікселями, відсотками, негативними значеннями.

Поєднують із параметрами position: relative, position: absolute, position: fixed та position: sticky (тільки top або bottom).

Властивість position використовують додатково із допоміжною властивістю z-index – яка керує накладанням один на одного елементів по осі Z. Кожен елемент може перебувати як нижче, так і вище за інші об'єкти веб-сторінки, їх розміщенням по z-осі керує властивість z-index. Ця властивість працює тільки для елементів, у яких значення position задано як absolute, fixed чи relative. Значення z-index задають простими цифрами, за замовчуванням значення рівне 0.

Властивість position може приймати наступні значення:

- static

Елементи відображаються як завжди. Використання властивостей left, top, right, bottom і z-index не призводить до результатів. Встановлюється за замовчування, тому його можна не задавати.

- relative

Положення елемента встановлюється відносно його початкового положення в коді. Додавання властивостей left, top, right та bottom змінює позицію елемента і зсуває його в той чи інший бік від початкового розташування.

Наприклад, для блоку_2, по-горизонталі ми зробимо відступ від лівої межі, а по-вертикалі – відступ від верхньої межі. Для цього необхідно задавати значення властивостей left та top.

Одночасне поєднання left та right, або top та bottom – неможливе.

```
.block_2 {
  background-color: aquamarine;
  position: relative;
  left: 50px;
  top: 60px;
}
```

Приклад. Є два блоки. Додамо властивість z-index із значенням 0 для блоку_2 та із значенням 1 для блоку_3, то ми побачимо, що блок_3 перекриває блок_2, тобто блок_3 на даний момент буде важливіший.

```
.block_2 {
  background-color: aquamarine;
  position: relative;
  left: 50px;
  top: 60px;
  z-index: 0;
}
.block_3 {
  background-color: blueviolet;
  position: relative;
  z-index: 1;
}
```

Якщо вказати для блоку_2 властивість z-index із значенням 2, то відбудеться перекриття блоком_2 блоку_3, і наданий момент, вже блок_2 буде вважатись головним.

- absolute

Елемент «нібито» виривається з коду та «зависає» у повітрі. Він перестає впливати на інші елементи. Додавання властивостей left, top, right і bottom змінює позицію елемента та зсуває його в ту чи іншу сторону відносно елемента-батька у якого заданий параметр position: relative.

Приклад. Для блоку_2 задаємо дану властивість.

```
.block_1 {
  background-color: darkkhaki;
}
.block_2 {
  background-color: aquamarine;
  position: absolute;
  /*розташування у верхньому лівому куті*/
  left: 0;
  top: 0;
}
.block_3 {
  background-color: blueviolet;
}
```

Що ми бачимо: блок_2 змістився на визначену нами позицію (верхній лівий кут вікна), в нього пропала властивість «ширина» (а ми знаємо, що у блочних елементів ширина 100%), він почав себе вести як рядковий елемент (його ширина рівна тепер ширині його контенту). Крім того, у батьківському блоці, його місце зайняв блок_3, який знаходився під ним.

Тому і говорять, що блок «нібито» виривається з коду та «зависає» у повітрі на тій позиції, яку ми йому вказуємо. І одночасно зникає простір, який цей блок (блок_2) займав до цього часу. Тобто відбулось підтягування всього контенту знизу догори в батьківському блоці.

Питання: в чому смисл застосування «абсолютного» позиціонування, та як зробити так, щоби можна було позиціонувати «абсолютний» елемент відносно іншого елемента?

Якраз із «абсолютним» позиціонуванням будуються більшість елементів. Проте воно спрацьовує тільки тоді, коли «абсолютне» позиціонування позиціонують відносно елемента-батька.

Для того, щоби блок_2 ми могли позиціонувати відносно елемента-батька (а це наш тег div із класом block), нам елементу-батьку, відносно якого ми хочемо позиціонувати наш «абсолютний» елемент, необхідно додати параметр position: relative. Цей параметр заставлятиме наш блок_2 позиціонуватись відносно його елемента-батька.

```
.block {
  border: 5px solid red;
  max-width: 800px;
  /*хоча й margin - це зовнішній відступ, проте із значеннями "0px auto" - цей параметр заставляє*/
  /*блочний елемент, який обмежений по ширині, вишикуватись по центру браузера*/
  margin: 0px auto;
  height: 800px;
  position: relative;
}
.block div{
  line-height: 50px;
  font-size: 18px;
  font-weight: 700;
  text-align: center;
  color: white;
}
.block_1{
  background-color: darkkhaki;
}
.block_2{
  background-color: aquamarine;
  position: absolute;
  /*розташування у верхньому лівому куті*/
  left: 0;
  top: 0;
}
.block_3{
  background-color: blueviolet;
}
```

І тепер замість верхнього кута браузера наш блок_2 перемістився у верхній лівий кут елемента-батька.

Для того, щоби «абсолютному» елементу вернути його початковий розмір, необхідно цей розмір задавати заново.

```
.block_2 {
  background-color: aquamarine;
  position: absolute;
  width: 100%;
  /*розташування у верхньому лівому куті*/
  left: 0;
  top: 0;
}
```

Тоді наш «абсолютний» елемент буде «зависати» по всій ширині елемента-батька та буде «перекривати» все, що знаходиться під ним на цій позиції.

Також із параметром `position: absolute` застосовна властивість `z-index`.

```
.block_2 {
  background-color: aquamarine;
  position: absolute;
  width: 100%;
  /*зсув на визначену позицію*/
  left: 20px;
  top: 30px;
}
.block_3 {
  position: relative;
  z-index: 2;
  background-color: blueviolet;
}
```

В блоці `_2` не встановлена властивість `z-index`, тому що вона враховується за замовчуванням із значенням 0.

- `fixed`

Елемент «нібито» виривається з коду та «зависає» у повітрі. Він перестає впливати на інші елементи. Фіксується, тобто не скролиться разом із вмістом. Додавання властивостей `left`, `top`, `right` і `bottom` змінює позицію елемента та зсуває його в ту чи іншу сторону тільки відносно вікна браузера, незалежно від того, чи є у елемента-батька параметр `position: relative` чи його немає.

```
.block_2 {
  background-color: aquamarine;
  position: fixed;
  top: 0;
  left: 0;
}
```

Також блок `_2` стає рядковим, і керування його шириною та висотою здійснюється через задання відповідних властивостей:

```
.block_2 {
  background-color: aquamarine;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
}
```

Також із параметром `position: fixed` застосовна властивість `z-index`.

```
.block_2 {
  background-color: aquamarine;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
}
.block_3 {
  position: relative;
  z-index: 2;
  background-color: blueviolet;
}
```

Блок_2 зафіксований, проте його буде покривати блок_3, оскільки в блоку_3 властивість z-index має більше значення.

- sticky («липкий» блок)

Застосування параметра position: sticky – це гібридне поєднання значень static та fixed.

Коротко: Спочатку елемент веде себе як static але після того як скрол доходить до його границі він стає fixed та прилипає до зазначеної позиції top або bottom.

Детально: Елемент позиціонується як static, але коли верхня границя елемента буде знаходитися на відстані, вказаній у параметрі top, від верхньої границі вікна браузера (або вказаному у параметрі bottom від нижньої), він залишиться в цьому положенні щодо вікна доти, доки його нижня границя не упреться в інший sticky елемент або кінець батьківського елемента.

Приклад. Наш блок_2 перетвориться із static у fixed в той момент, коли при скролінгу верхня границя браузера досягне його положення, тобто блок_2 «прилипає» до верхньої границі браузера і його дії подібні при параметру position: fixed, однак в межах батька-елемента, а не вікна браузера.

```
.block_2 {
  background-color: aquamarine;
  position: sticky;
  top: 0;
}
```

Приклад. Вказуємо відстань в пікселях, яка повинна бути між верхньою границею браузера та нашим блоком_2.

```
.block_2 {
  background-color: aquamarine;
  position: sticky;
  top: 20px;
}
```

4. CSS псевдокласи

CSS псевдокласи – найчастіше використовуються для застосування стилів залежно від стану елемента чи його положення в коді.

Розрізняють псевдокласи: hover, active, visited, focus, first-child, last-child, nth-child.

Синтаксис запису псевдокласу:

```
селектор:псевдоклас {
    параметри стилю;
}
```

Розрізняють псевдокласи стану та розташування в коді.

Псевдокласи стану.

- `:hover`

Дозволяє задавати CSS-стилі при наведенні курсору миші на HTML-елемент, при цьому активація HTML-елемента не обов'язкова.

```
.link:hover{
  text-decoration: underline;
  color: blue;
}
```

Також псевдо-клас `:hover` активно використовується не тільки для посилань, так і для будь-якого іншого елемента.

Наприклад, змінити фон блоку `div` на зелений при наведенні на нього.

```
.block:hover{
  background-color: green;
}
```

- `:visited`

Спрацьовує для відвідуваних посилань.

```
.link:visited{
  color: green;
}
```

- `:active`

За допомогою псевдокласу `:active` ми можемо задавати CSS-стилі в момент натискання на HTML-елемент мишею. В основному застосовується до посилань та кнопок.

```
.link:active{
  color: yellow;
}
```

- `:focus`

Псевдо-клас `:focus` дозволяє задавати стилі для HTML-тегів таких як `<input>`, `<textarea>`, `<checkbox>` та інших при встановленні курсору миші цей елемент.

```
.textinput:focus{
  box-shadow: 0 0 15px rgba(0, 0, 0, 0.75);
  background-color: rgb(241, 227, 227);
}
```

Щоб стилізувати посилання належним чином, необхідно дотримуватись LVHA-порядку застосування псевдокласів `:link -> :visited -> :hover -> :active`.

Псевдоклас `:focus` зазвичай розміщується прямо перед або відразу після `:hover`, залежно від очікуваного ефекту.

Псевдокласи положення в коді.

- `:first-child`

Звернення до першого елемента у блоці.

```
.list li:first-child{
  color: green;
}
```

- `:last-child`

Звернення до останнього елемента у блоці.

```
.list li:last-child{
  color: blue;
  font-weight: bold;
}
```

- :nth-child(порядковий номер елемента)

Звернення до конкретних елементів у блоці.

Значення:

- (порядковий номер елемента) – звертається до елемента, порядковий номер якого задається (нумерація стартує із числа 1)

```
.list li:nth-child(3){
  color: purple;
  font-style: italic;
}
```

- (odd) – звертається до елементів з непарними номерами

```
.list li:nth-child(odd){
  text-decoration: underline;
}
```

- (even) – звертається до елементів із парними номерами

```
.list li:nth-child(even){
  text-transform: uppercase;
}
```

Так само ми можемо звернутись і до інших html-елементів, наприклад до тегу <p>.

```
.block p:first-child{
  font-style: italic;
}
.block p:last-child{
  text-transform: uppercase;
}
.block p:nth-child(3){
  font-weight: 700;
}
```

- (An+B) – звертається до елементів індекси яких, визначаються за формулою функціонального запису, де A,B – цілі числа, n>=0 (стартує від 0).

```
.list li:nth-child(3n+3){
  background-color: aqua;
}
```

5. CSS псевдоелементи

Псевдоелементи можуть за допомогою CSS звернутись до вмістимого чи частини вмістимого об'єкта, а також додати до об'єкта нові елементи.

Розрізняють псевдоелементи: before, after, first-line та first-letter.

Синтаксис запису псевдоелемента:

```
селектор::псевдоелемент {
  параметри стилю;
}
```

- ::first-line

Задає стиль першого рядка форматowanego тексту.

```
.text::first-line{
  color: red;
}
```

У правилах стилю даного псевдоелементу припустимо використовувати лише властивості, що стосуються шрифту, зміни кольору тексту та фону.

- `::first-letter`

Визначає стиль першого символу в тексті елемента, до якого додається.

```
.text::first-letter{
  color: blue;
  font-size: 50px;
}
```

До цього псевдоелементу можуть застосовуватися лише стиліові властивості, пов'язані з властивостями шрифту, полями, відступами, границями, кольором та фоном.

- `::before`

Застосовується для відображення бажаного контенту до відображення вмістимого елемента, до якого він додається. Обов'язково поєднують `::before` із властивістю `content`, без якої `::before` не буде відображатись в браузері.

Властивість `content` може містити:

- порожнє значення: `content: ''`;
- необхідний контент, який ми хочемо вивести до відображення вмістимого нашого блоку.

При додаванні `::before` до блокового елемента значення властивості `display` може бути тільки: `block`, `inline`, `none`, `list-item`. Всі інші значення будуть трактуватись як `block`.

При додаванні `::before` до рядкового елемента, `display` обмежений значеннями `inline` та `none`. Всі інші значення сприйматимуться як `inline`.

- `::after`

Використовується для виведення бажаного контенту після відображення вмістимого елемента, до якого він додається. Псевдоелемент `::after` працює також разом із властивістю `content`.

При додаванні `::after` до блокового елемента, значення властивості `display` може бути лише: `block`, `inline`, `none`, `list-item`. Всі інші значення будуть трактуватись як `block`.

При додаванні `::after` до рядкового елемента, `display` обмежений значеннями `inline` та `none`. Всі інші сприйматимуться як `inline`.

Приклад 1.

```
.list li::before{
  content: '#';
}
.list li::after{
  content: ' ';
}
```


Приклад 2.

```
.list li::before{
  content: '';
  width: 10px;
  height: 10px;
  border-radius: 50%;
  background-color: ■ brown;
  display: inline-block;
  margin-right: 15px;
}
```

Даний приклад демонструє, що без застосування будь-яких фонів, картинок, background-ів, за допомогою простого псевдоелемента можна додавати до вмістимого контенту елемента нові елементи, які ми можемо стилізувати під свої потреби.

Найчастіше псевдоелементи ::before та ::after використовуються в поєднанні із абсолютним позиціюванням.

Псевдоелементи можна комбінувати із псевдокласами, проте записувати їх потрібно у відповідному порядку:

```
.list li:hover::before{
  background-color: ■ yellow;
}
```

ПРАКТИЧНА ЧАСТИНА


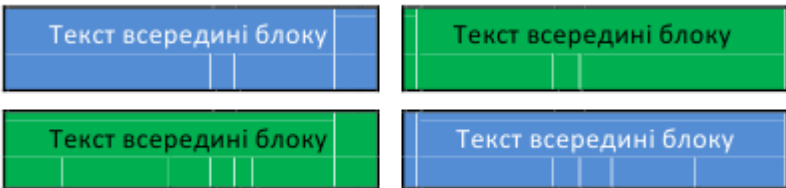
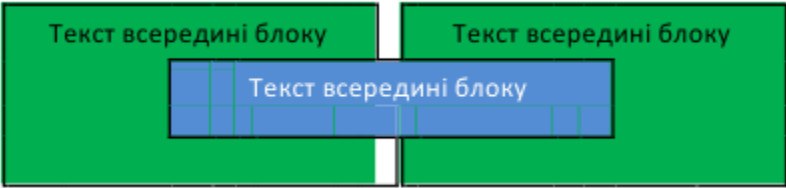
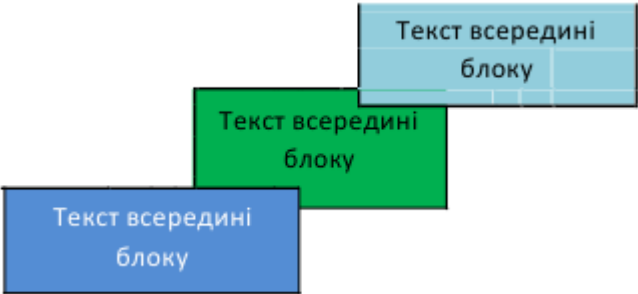
Завдання 1

Відповідно до свого варіанту створіть HTML-документ та за допомогою CSS відформатуйте блочні елементи у ньому таким чином, щоб він став схожий на зображення в табл.1.

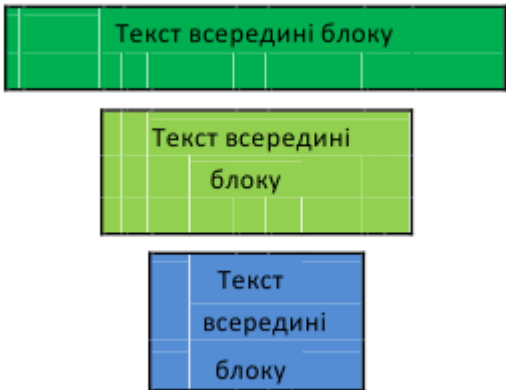

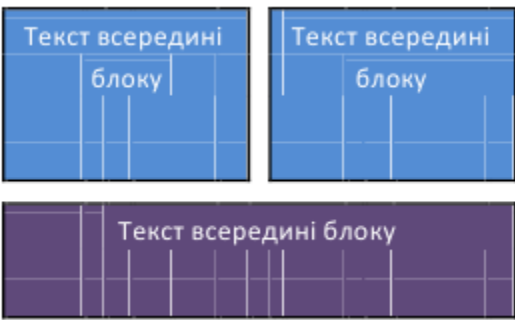
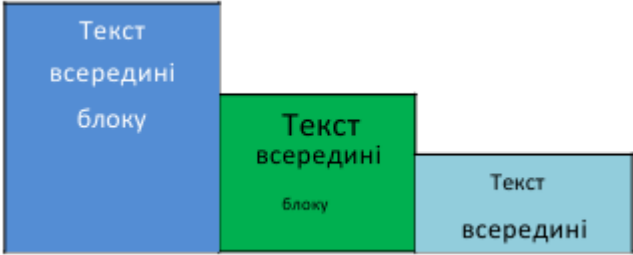
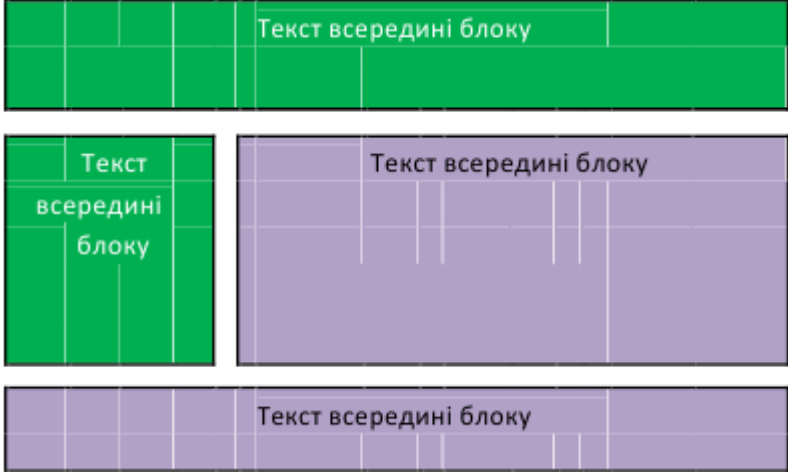
Примітка:

Для того, щоб відформатувати блоки таким чином, щоб один блок розміщувався над іншим вам необхідно використати CSS властивості (position: absolute; top: 20px; left: 50px;), або інші значення (top, left) відповідно до того, де саме повинні розміщуватися ваші блоки. А щоб один блок розмістити поверх іншого використовуйте властивість (z-index) і чим більше значення властивості (1, 2, 3), тим вище розміщуватиметься блок.

Таблиця 1 – Завдання для самостійного виконання студентом

№ варіанту	Завдання
1	
2	
3	
4	

5	
6	
7	
8	
9	
10	

11	
12	
13	
14	
15	

16	
17	
18	
19	

20	<div>Текст всередині блоку</div> <div>Текст всередині блоку</div> <div> <div>Текст всередині блоку</div> <div>Текст всередині блоку</div> </div>
----	--

Завдання 2


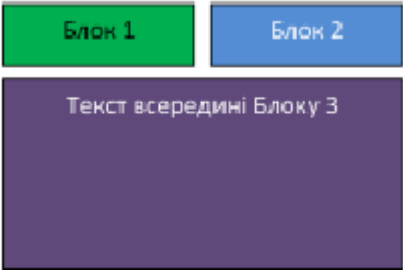
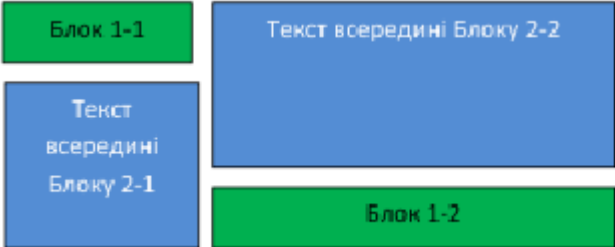
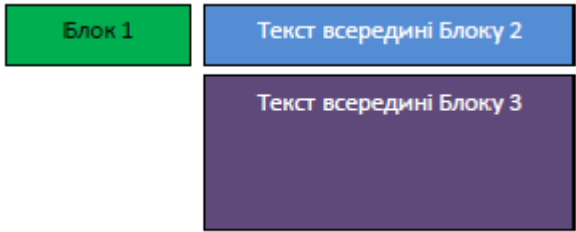
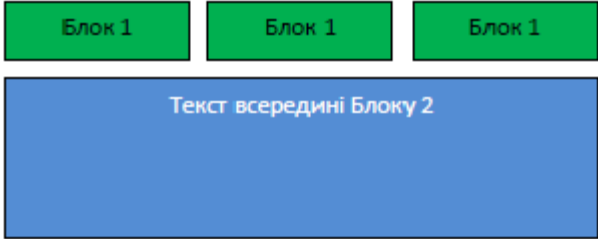
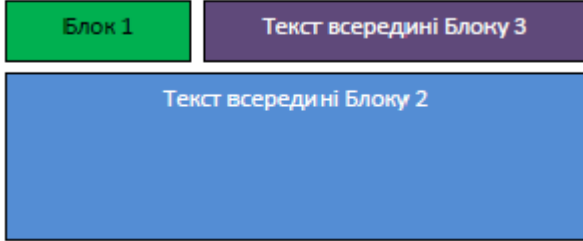
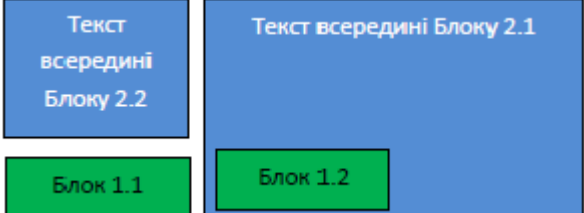
Відповідно до свого варіанту створіть HTML документ та за допомогою CSS реалізуйте поставлене завдання.

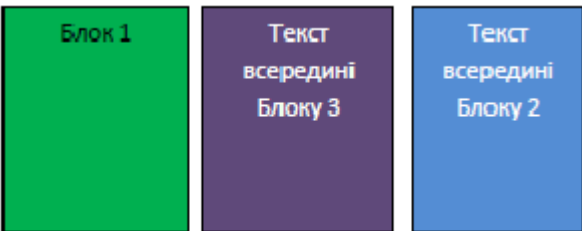
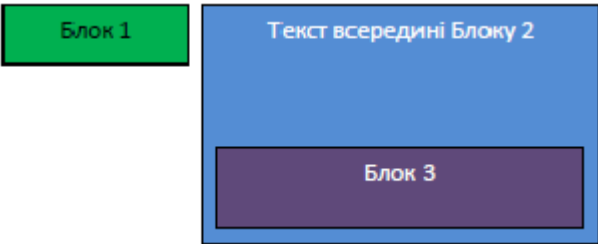
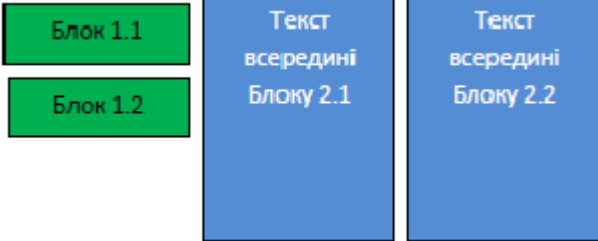
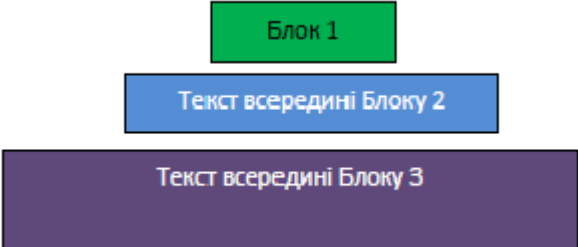
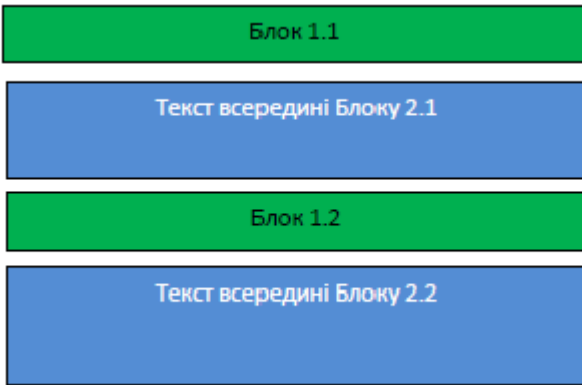
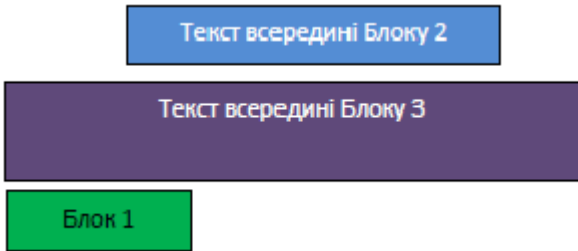
Примітка:


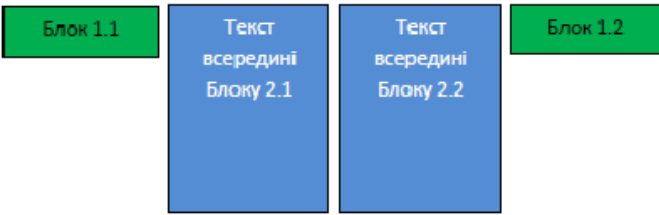

Для того, щоб відстежити поведінку курсору над певним елементом використовуйте псевдоклас :hover.

Таблиця 2 – Завдання для самостійного виконання студентом

№ варіанту	Завдання	Пояснення
1	<div>Блок 1</div> <div>Текст всередині Блоку 2</div> <div>Текст всередині Блоку 3</div>	Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.
2	<div>Блок 1.1</div> <div>Текст всередині Блоку 2.1</div> <div>Текст всередині Блоку 2.2</div> <div>Блок 1.2</div>	Необхідно щоби при наведенні на Блок 1.1 з'являвся Блок 2.1, а при наведенні на Блок 1.2 з'являвся Блок 2.2.
3	<div>Блок 1</div> <div>Текст всередині Блоку 2</div> <div>Блок 1</div>	Необхідно щоби при наведенні на Блок 1 (той що зліва) з'являвся Блок 2, а при наведенні на Блок 1 (той що справа) зникав Блок 1 (той що зліва).
4	<div>Блок 1</div> <div>Текст всередині Блоку 2</div> <div>Текст всередині Блоку 3</div>	Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.

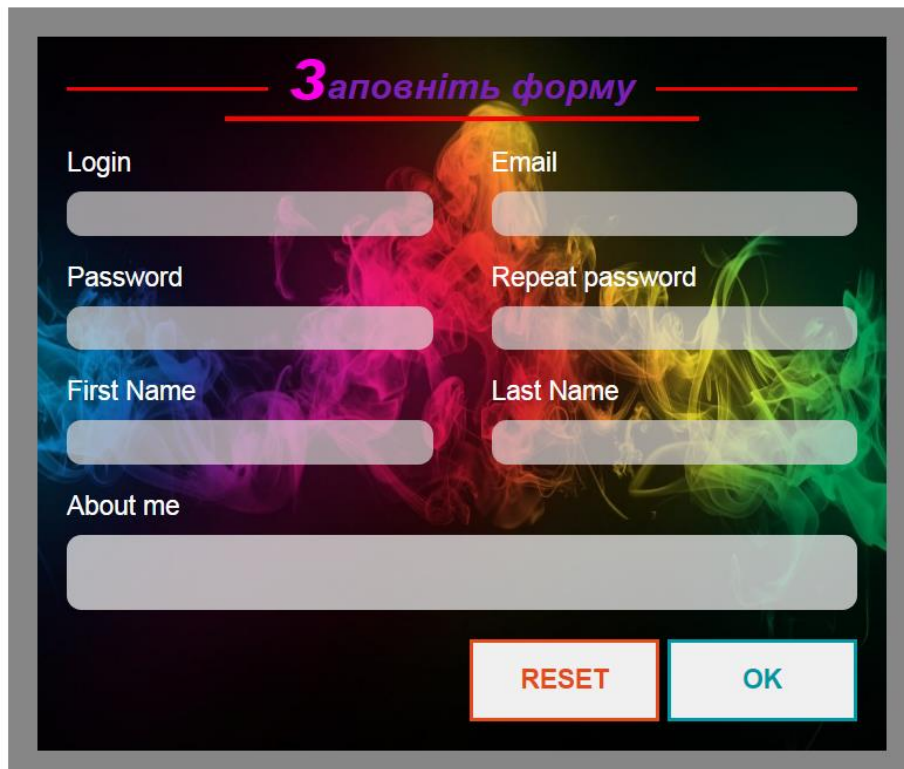
5		Необхідно щоби при наведенні на Блок 1 (будь-який) з'являвся Блок 2, знаходячись над Блоком 1.
6		Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.
7		Необхідно щоби при наведенні на Блок 1.1 з'являвся Блок 2.1, а при наведенні на Блок 1.2 з'являвся Блок 2.2.
8		Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.
9		Необхідно щоби при наведенні на Блок 1 (будь-який) з'являвся Блок 2.
10		Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.
11		Необхідно щоби при наведенні на Блок 1.1 з'являвся Блок 2.1, а при наведенні на Блок 1.2 з'являвся Блок 2.2.

12		Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.
13		Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.
14		Необхідно щоби при наведенні на Блок 1.1 з'являвся Блок 2.1, а при наведенні на Блок 1.2 з'являвся Блок 2.2.
15		Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.
16		Необхідно щоби при наведенні на Блок 1.1 з'являвся Блок 2.1, а при наведенні на Блок 1.2 з'являвся Блок 2.2.
17		Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.

18		Необхідно щоби при наведенні на Блок 1 з'являвся Блок 2, а при наведенні на Блок 2 з'являвся Блок 3.
19		Необхідно щоби при наведенні на Блок 1.1 з'являвся Блок 2.1, а при наведенні на Блок 1.2 з'являвся Блок 2.2.
20		Необхідно щоби при наведенні на Блок 1.1 з'являвся Блок 2.1, а при наведенні на Блок 1.2 з'являвся Блок 2.2.

Завдання 3

Реалізувати форму, зовнішній вигляд якої зображено на рисунку через застосування псевдокласів та псевдоелементів.



Завантажте форму

Login

Email

Password

Repeat password

First Name

Last Name

About me

RESET OK

Ми вдячні вам

Login

Email

Password

Repeat password

First Name

Last Name

About me

RESET **OK**

Текст

- Шрифт: сімейство: Arial, "Helvetica Neue", Helvetica, sans-serif
- Колір:
 - label – white;
 - input, textarea – black;
- Розмір кеглю:
 - label, button – 18px;
 - input, textarea – 16px;

Форма

- Ширина форми – 620px, колір границі – #868686, товщина границі – 20px, задній фон – картинка, внутрішні відступи: зверху – 0px, справа, зліва, знизу – 20px.

Джерела картинок:

<https://phonoteka.org/57557-fon-dlja-oprosa.html>

<https://funart.pro/24730-krutoj-fon-64-foto.html>

- Поля форми: для введення тексту та пошти, для введення паролю, для введення багаторядкового тексту, кнопки для відправки та для очищення.

Заголовок форми

- Розташування тексту – по центру, розмір кеглю – 25px, колір тексту – #7d21b3, насиченість шрифту (вага шрифту) – 700, зображення шрифту – курсивне, зовнішні відступи – 20px.

Елементи форми


- До полів для введення тексту/ багаторядкового тексту, пошти та паролю застосовано: насиченість шрифту (вага шрифту) – 700, величина абзацного відступу – 20px, заокруглення кутів елемента – 10px, рівень прозорості елемента – 0.6, відсутність внутрішньої та зовнішньої границь навколо елемента.

- До поля для введення багаторядкового тексту застосовано: ширина – 540px, висота – 46px, заборона на зміну розмірів елемента, внутрішні відступи – 5px.
- До полів для введення тексту, пошти та паролю застосовано: ширина – 250px, висота – 30px, перетворення елемента до блочного вигляду. Зовнішні відступи підібрати самостійно. Розташування полів реалізоване через властивість float-позиціонування в поєднанні з псевдо-елементом ::after, доданого до батьківського елемента:

Приклад:

```
.parent::after {
  display: block;
  content: '';
  clear: both;
}
```

Кнопки

- Ширина – 130 пікселів, шрифт – жирний, у верхньому регістрі, міжрядковий інтервал тексту – 50px. Форма курсору, коли він перебуває у межах елемента (pointer ).
- Кольори:
 - кнопка для очищення – border: 3px #e64a19, text: #e64a19;
 - кнопка для відправки – border: 3px #0097a7, text: #0097a7;

Значення елементів форми при використанні псевдокласів:

Елемент	:hover	:focus	:active
Поля для введення тексту та пошти	нижня внутрішня границя – 4px, колір – #fbff00	нижня внутрішня границя – 4px, колір – #ffa600	
Поле для введення паролю	нижня внутрішня границя – 4px, колір – #1eff00	нижня внутрішня границя – 4px, колір – #00ffea	
Поле для введення багаторядкового тексту	тінь елемента: радіус розмиття тіні – 10px, розтягнення тіні – 10px, колір тіні – rgba(233, 27, 154, 0.6)		
Кнопка для відправки	border: 3px #0097a7, text: white; background-color: #0097a7, border-radius: 15px		border: 3px #ecfd00, background-color: #014950, border-radius: 25px, position: relative

Кнопка для очищення	border: 3px #e64a19, text: white, background-color: #e64a19, border-radius: 15px		border: 3px #ecfd00, background-color: #a02904, border-radius: 25px, position: relative
---------------------	--	--	---

Застосування до елементів форми та кнопок псевдоелементів:

- Поле для введення багаторядкового тексту:
 - `::first-line` – робить текст першого рядка червоним.
- Заголовок форми:
 - `::first-letter` – задає для першої літери тексту розмір кеглю – 50px, колір тексту – pink.
 - `::before` та `::after` в поєднанні із `position: relative`; та `position: absolute`; для додавання ліній перед та після заголовку форми.
 - `::after` – для додавання лінії під заголовком форми.
- Комбінація псевдокласу `:hover` із псевдоелементами `::before` та `::after` для заміни назви заголовку форми та стилів лінії під заголовком форми.
- Кнопки – комбінація псевдокласу `:active` із псевдоелементом `::after` для відображення піктограми.

Завдання 4

Реалізувати навігаційні панелі (горизонтальну та вертикальну) із випадними пунктами меню.

Використати багаторівневі марковані списки, елементами яких будуть посилання. Колір, шрифт, розміри та інші властивості підібрати самостійно.

В даному завданні необхідно використовувати властивості, що пов'язані з позиціонуванням.

Передбачити:

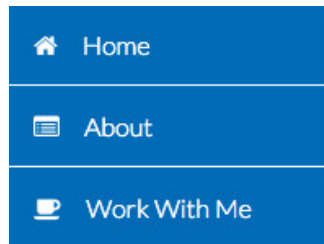
- Для обох панелей навігації:
 - відображення активного (поточного) пункту меню.
 - при наведенні на пункти меню, вигляд навігації змінюється (`:hover`).
- Для горизонтальної навігаційної панелі:
 - перший рівень навігації повинен бути представленим піктограмами (панель значків) (піктограми – <https://fonts.google.com/icons>).

Наприклад:



- фіксацію (`fixed` позиціонування) панелі навігації у верхній частині сторінки.
- вирівнювання останнього пункту меню справа сторінки через використання плаваючих елементів.
- створення (`sticky`) «липкої» навігаційної панелі меню.
- Для вертикальної навігаційної панелі:
 - Перший рівень навігації повинен бути представленим піктограмами (панель значків) (піктограми – <https://fonts.google.com/icons>) та назвою пункту меню.

Наприклад:



- повна висота фіксованої вертикальної навігації.
- передбачити три рівня вкладеності.

Завдання 5

Реалізувати HTML сторінку за прикладом. Використовуючи CSS властивості для роботи з тінню, прозорістю та відображенням елементів.

