Лабораторна робота 1.3.

Застосування каскадних таблиць стилів CSS. Блокова модель CSS

Мета: Ознайомитися з методами застосування каскадних таблиць стилів та особливостями блокової моделі CSS.

Вимоги до обладнання та програмного забезпечення

Лабораторна робота виконується на ПК з використанням програм Microsoft Expression Web 4, TopStyle 4 та браузерів Google Chrom, Opera, Mazila Firefox.

Основні теоретичні відомості

CSS *(Cascading Style Sheets)* – мова таблиць стилів, яка дозволяє прикріплювати стиль (наприклад, шрифти і колір) до структурованих документів (наприклад, документів HTML і додатків XML). Зазвичай CSS-стилі використовуються для створення і зміни стилю елементів веб-сторінок і призначених для користувача інтерфейсів, написаних мовами HTML *і* XHTML, але також можуть бути застосовані до будь-якого виду XML-документа, зокрема й XML, SVG *і* XUL. Відокремлюючи стиль подання документів від вмісту документів, CSS спрощує створення веб-сторінок і обслуговування сайтів.

Існують такі способи підключення стилів:

1. Зв'язування таблиці стилів з документом *(linking)*. Підключення CSS-файла відбувається за допомогою елемента *link*. Зовнішня таблиця стилів становить собою текстовий файл, який містить опис стилів:
   <link rel="stylesheet" type="text/css" href="style.css" />
2. Вкладення стилів *(embeding)*. Інформація про стилі міститься в заголовку веб-сторінки в елементі *style*.
   <head>
   <title>Назва документа</title>
   <style>
   h1{color:red; font-style:italic; font-size:32px}
   </style>
   </head>
3. Вбудовування стилів *(inline)*. Стиль поміщається всередину тега за допомогою атрибута *style*.
   <h1 style="font-size:40px; color:red; text-align:center">Заголовок розділу</h1>
4. Імпорт стилів *(import)*
   <head>
   <title>Назва </title>
   <style type="text/css">
   @import: url(mystyle.css);
   </style>
   </head>

Основним поняттям у CSS є селектор – це ім'я стилю, для якого додаються параметри форматування. Як селектор виступають елементи розмітки, класи й ідентифікатори. Загальний спосіб запису має такий вигляд:

селектор {властивість: значення;}

Спочатку вводять ім'я селектора, наприклад, *<h1>*. Це означає, що всі стильові параметри застосовуватимуть до елемента *<h1>*, потім йдуть фігурні дужки, в яких записують стильову властивість, а його значення вказують після двокрапки. Стильові властивості розділяють між собою крапкою з комою, в кінці цей символ можна не писати.

Приклад 1.22

```
<html>
<head>
<style type="text/css">
<!--
body {background-color: #9999cc;}
h1 {background-color: green;
font-size: 200%;
color: white;
text-align: center;}
-->
</style>
</head>
```

```
<body>
<h1> Синтаксис CSS </h1>
</body>
</html>
```

Коли потрібно застосувати однакові стилі для декількох елементі тоді слід перед фігурними дужками перелічити їх назви. Отримаємо груповий селектор: *p, h1 {color: darkred;}*.

Існують ще два способи визначення стилів – а саме через класи й ідентифікатори.

Припустимо, нам необхідно задати властивості елемента *<p>*, але кожен абзац повинен відрізнятися від попереднього. Досягти даної мети відомими нам способами неможливо, ось тут і приходять на допомогу класи.

Приклад 1.23

```
<html>
<head>
<style type="text/css">
<!--
p.one {background-color: #d6d2dd; font-style: regular; font-size: 15;}
p.two {background-color: #d1ded7; font-style: bold; font-size: 20;}
p.three {background-color: #ddd8d2; font-style: italic; font-size: 25;}
-->
</style>
</head>
<body>
<p class="one">CSS має дуже простий синтаксис.
<p class="two">Знаючи CSS, можна створити якісні сайти
<p class="three"> Каскадні таблиці стилів являють собою опис різних елементів HTML і створені вони для розширення властивостей останніх. Вперше стилі були запропоновані WWW Consorcium у рамках розробки специфікації HTML 3.0.
</body>
</html>
```

Коли треба створити клас, не прив’язаний до певного елементу, тоді визначення елемента потрібно опустити.

Приклад 1.24

```
<html>
<head>
<title>Селектор клас</title>
<style type="text/css">
.one {color: green;}
.two {color: blue;}
</style>
</head>
<body>
<div class="one">Текст відображається зеленим кольором.
<p class="two">Абзац відображається синім кольором.
<hr class="one">
</body>
</html>
```

Якщо потрібно виділити один елемент, унікальний, відмінний від всіх інших у документі, тоді можна використати ідентифікатор.

Приклад 1.25

```
<html>
<head>
<title>Селектор ідентифікатор</title>
<style>
#firstheader {
color: red; // задає червоний колір тексту
```

```
font-weight: bold; // шрифт стане жирним
text-align: center; // центрування
}
</style>
</head>
<body>
<h1 id="firstheader"> Перший заголовок на сторінці </h1>
</body></html>
```

Для форматування символів за допомогою CSS використовують властивості *font-family, font-size, font-style, font-variant, font-weight.*

font-family – визначає сімейство шрифтів. Можна задавати список шрифтів, у такому разі останній елемент списку – назва сімейства.

Основні сімейства шрифтів:

- *serif* – шрифти з зарубками (приклад: *Times New Roman*);
- *sans-serif* – шрифти без зарубок (приклади: *Arial, Verdana*);
- *cursive* – курсивні шрифти (приклад: *Comic Sans MS*);
- *fantasy* – декоративні шрифти (приклад: *Monotype Corsiva*);
- *monospace* – шрифт фіксованої ширини, ширина кожного символу в такому сімействі однакова (приклад: *Courier New*).

Якщо назва шрифту складається з двох або більше слів, її треба писати в лапках: *p {font-family: "Times New Roman", Times, serif; }.*

font-size – задає розмір шрифту. Абсолютні значення розміру шрифту задаються як *xx-small, x-small, small, medium, lagre, x-large, xx-large*. Відносні задаються як *larger, smaller*, або у відносних значеннях довжини, наприклад *em* (висота шрифту елемента), *ex* (висота символу *x*), пункти *(pt),* пікселі *(px)*, відсотки *(%)* та ін. (За 100 % береться розмір шрифту батьківського елемента).

font-style – визначає стиль шрифту з обраного сімейства: нормальний *(normal)*, курсивний *(italic)* або похилий *(oblique)*, який генерується з нормального невеликим нахиленням символів.

font-variant – визначає, як потрібно зображати малі літери – залишити їх без модифікацій чи робити їх всі великими зменшеного розміру. Такий спосіб зміни символів називається капітеллю: *normal* і *small-caps.*

font-weight – визначає жирність шрифту із заданого сімейства. Може задаватися числами 100, 200, ..., 900 або ключовими словами: *extra-light, light, demi-light*, *medium (normal)*, *demi-bold, bold, extra-bold*. Гарантовано підтримуються лише *normal (400), bold (700)*. Може визначатися відносно: *bolder, lighter*.

Можна задати всі властивості шрифту одночасно. Синтаксис:

font: font-style font-variant font-weight font-size/line-height font-family: p {font: italic small-caps 600 18pt/24pt "Arial, sans-serif"}.

Якщо якесь значення не задане, використовується відповідне значення за замовчуванням. Усі властивості шрифтів успадковуються вкладеними елементами і можуть застосовуватися до всіх елементів HTML.

Блокова модель CSS описує прямокутний блок, створюваний для елемента в дереві документа і виводиться згідно візуальної моделі форматування (специфікація W3C). Це означає, що елемент у HTML-документі виводиться всередині свого окремого прямокутного блока.

У специфікації HTML4.01 за замовчуванням до блочних елементів відносяться *address, blockquote, div, dl, fieldset, form, hr, h1-h6, noscript, ol, p, pre, table, ul.*

Типовим блоковим елементом HTML є елемент розмітки *div*. У елемента розмітки *div* доступні всі універсальні атрибути і події HTML, унікальних атрибутів у елемента розмітки *div* немає, вміст елемента *div* оформляється за допомогою таблиць стилів CSS.

Блок складається з безпосередньо контенту (внутрішнього вмісту), внутрішніх відступів, кордонів і, нарешті, зовнішніх відступів.

Для управління кожною із складових частин блока існують відповідні CSS-властивості: внутрішні відступи – *padding*, межі - *border*, зовнішні відступи – *margin*. За бажання ці властивості можна задати для кожної сторони блока окремо (рис. 1.1):

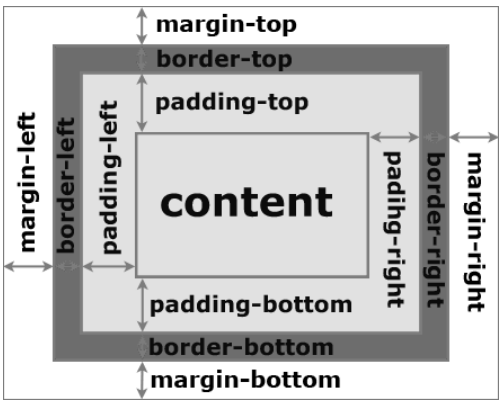

Рис. 1.1. Властивості блока

Приклад 1.26:

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset=utf-8">
<title>margin</title>
<style type="text/css">
BODY {
margin: 0; /* Прибираємо відступи */
}
div.big {
margin: 20%; /* Відступи навколо елемента */
background: #fd0; /* Колір фону */
padding: 10px; /* Поля навколо тексту */
}
div.small {
border: 3px solid #666; /* Параметри рамки */
padding: 10px; /* Поля навколо тексту */
margin: 10px; /* Відступи навколо */
}
</style>
</head>
<body>
<div class="big">
<div class="small">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh
euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim
ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl
ut aliquip ex ea commodo consequat.
</div>
</div>
</body>
</html>
```

Для елемента можна задати межі (рамку) і два види відступів. Чим же вони відрізняються?

Основна відмінність відразу впадає в очі: *padding* – це відступ між контентом і межею, а *margin* – це відступ між межею і «зовнішнім світом».

Звідси випливає друга відмінність – якщо для елемента задати фон *(background),* то цим фоном заллється і контент, і внутрішній відступ *(padding). Margin,* перебуваючи зовні, завжди залишається прозорим.

Padding, якщо провести аналогію – це поля на аркуші паперу. Вони мають той самий колір, що і лист, але текст на них не заходить. *Margin* – це відстань від краю аркуша до іншого краю, що лежить поруч листа або, скажімо, до краю столу.

Третя відмінність полягає в тому, що *padding* та *margin* по-різному беруть участь у підрахунку загальної ширини блока. Це залежить від прийнятої блочної моделі.

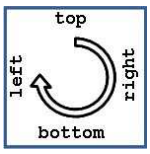

Рис. 1.2. Порядок встановлення параметрів відступів

Для браузера кожен елемент розмітки – це контейнер, у якого є вміст, внутрішній відступ, зовнішні поля, а також рамка. Блок займає простір на сторінці, що дорівнює сумі ширини вмісту, відступу, поля і рамки.

Margin – властивість для встановлення ширини полів для всіх сторін елемента або окремо. У цієї властивості може бути від одного до чотирьох значень. Якщо є лише одне значення, воно буде присвоєно відразу до всіх полів. Якщо два значення, перше з них присвоюється верхньому і нижньому полю, а друге – лівому і правому. Якщо ж три, то перше значення присвоюється верхньому полю, друге – лівому і правому, а третє – нижньому (рис. 1.2).

Можливі атрибути:

- *margin-top* – ширина верхнього поля;
- *margin-right* – ширина правого поля;
- *margin-bottom* – ширина нижнього поля;
- *margin-left* – ширина лівого поля;
- *inherit* – застосовується значення батьківського елемента.

Padding – властивість для встановлення ширини відступів відразу для всіх сторін елемента або окремо. У цієї властивості може бути від одного до чотирьох значень. Якщо є лише одне значення, воно буде присвоєно відразу всім полям. Якщо два значення, перше з них присвоюється верхньому і нижньому полю, а друге – лівому і правому. Якщо ж три, то перше значення присвоюється верхньому полю, друге лівому і правому, а третє – нижньому.

Можливі атрибути:

- *padding-top* – ширина верхнього відступу;
- *padding-right* – ширина правого відступу;
- *padding-bottom* – ширина нижнього відступу;
- *padding-left* – ширина лівого відступу;
- *inherit* – застосовується значення батьківського елемента.

border – властивість для визначення межі відразу з усіх сторін елемента. Значення встановлюється однаковими для всіх його сторін.

Можливі атрибути:

- *border-width* – товщина межі;
- *border-style* – стиль межі;
- *border-color* – колір межі;
- *inherit* – застосовується значення батьківського елемента.

Рис. 1.3. Типи границі блока

border-width – властивість для встановлення ширини межі відразу для всіх сторін елемента або окремо. У цієї властивості може бути від одного до чотирьох значень. Можливі атрибути:

- *border-top-width* – товщина верхньої сторони;
- *border-right-width* – товщина правої сторони;
- *border-bottom-width* – товщина нижньої сторони;
- *border-left-width* – товщина лівої сторони;
- *inherit* – застосовується значення батьківського елемента.

border-style – властивість для установки стилю межі відразу для всіх сторін елемента або окремо. У цієї властивості може бути від одного до чотирьох значень.

Можливі атрибути:

- *border-top-style* – стиль верхньої сторони.
- *border-right-style* – стиль правої сторони.
- *border-bottom-style* – стиль нижньої сторони.
- *border-left-style* – стиль лівої сторони.
- *inherit* – застосовується значення батьківського елемента.

border-color – властивість для установки кольору границь відразу для всіх сторін елемента. У цієї властивості може бути від одного до чотирьох значень.

Можливі атрибути:

- *border-top-color* – колір верхньої сторони.
- *border-right-color* – колір правого боку;
- *border-bottom-color* – колір нижньої сторони;
- *border-left-color* – колір лівої сторони;
- *inherit* – застосовується значення батьківського елемента.

Завдання до самостійної роботи

1. Виконайте приклади.
2. Відредагуйте сторінку, присвячену навчальній групі за допомогою стилів.
3. Розробіть персональну веб-сторінку, яка повинна включати вашу біографію використовуючи блокову модель таблиці стилів. Автобіографія повинна мати чіткий план і структуру (фото обов'язково).

Контрольні запитання та завдання

1. Назвіть призначення каскадних таблиць стилів.
2. Які ви знаєте способи використання стилів на веб-сторінках?
3. Як за допомогою CSS виконати відформатувати текст?
4. Назвіть особливості блокової моделі CSS.
5. Які існують відступи в блоковій моделі CSS?
6. У чому полягає призначення елемента розмітки *div*?

Лабораторна робота 1.4

Псевдокласи і псевдоелементи. Блокова верстка

Мета: ознайомитися з методами застосування псевдокласів і псевдоелементів та методами блокової верстки.

Вимоги до обладнання та програмного забезпечення

Лабораторна робота виконується на ПК з використанням програм Microsoft Expression Web 4, TopStyle 4 та браузерів Google Chrom, Opera, Mazila Firefox.

Основні теоретичні відомості

Псевдокласи визначають динамічний стан елементів, який змінюється за допомогою дій користувача. Прикладом такого стану може бути текстове посилання, яке змінює свій колір при наведенні на нього курсора миші. При використанні псевдокласів браузер не перенавантажує поточний документ, тому за допомогою псевдокласів можна отримати різні динамічні ефекти на сторінці.

Синтаксис застосування псевдокласів такий: *Селектор:Псевдокласс { Опис правил стилю }*

Псевдокласи застосовуються до імен ідентифікаторів або класів *(a.menu :hover {color: green})*, а також до контекстних селекторів *(.menu a :hover {background: #fc0})*. Якщо псевдоклас вказується без селектора попереду *(:hover)*, то він застосовуватиметься до усіх елементів документа.

Існують псевдокласи посилань. У цій категорії два псевдокласи *:link* і *:visited*. За їх допомогою задають стилі для елементів, що є посиланнями (під такими розуміються елементи, що мають атрибут *href*), і для тих посилань, на які користувач уже натискав.

:link – застосовується до посилань, які ще не відвідував користувач, і задає для них стильове оформлення;

:visited – для вже відвіданих посилань.

Приклад 1.27.

```
a:link {
   color: #036;
}
a:visited {
color: #80bd34;
}
```

Існують динамічні псевдокласи. Ці псевдокласи застосовуються до елементів залежно від дій користувача. *CSS2* визначає три такі псевдокласи *-:hover,:active і :focus*.

:hover – елемент, над яким розміщується покажчик миші. Щойно покажчик йде за межі елемента, стилі, задані цим псевдокласом, скасовується.

:active – елемент, активований користувачем (наприклад, посилання або кнопка у момент клацання).
:focus – елемент, якому належить фокус введення.

Приклад 1.28

```
a:hover
{
text-decoration: none;
background-color: #333099;
}
```

Використовуючи динамічні псевдокласи можна оживити веб-сторінки без використовування мови JavaScript.

Як селектор псевдокласу може виступати не лише який-небудь елемент – елемент розмітки, але й клас або ідентифікатор, наприклад: *.menu :hover { color:#ff0000;}*

Приклад 1.29

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="windows-1251">
  <title>Приклади застосування псевдокласів</title>
    <style>
   a:link {
    color: #036; /* Колір невідвіданих посилань */
    }
   a:visited {
    color: #606; /* Колір відвіданих посилань */
    }
   a:hover {
color: #f00; /* Колір посилань при наведенні на них курсору миші */
}
a:active {
 color: #ff0; /* Колір активних посилань  */
}
input:focus {
color: red; /* Червоний колір */
}
table { border-spacing: 0; }
td { padding: 4px; }
tr:hover {
background: #fc0; /* Змінює колір тла рядка таблиці */
}
</style>
</head>
<body>
<p>
<a href="1.html">Посилання №1</a> |
<a href="2.html">Посилання №2</a> |
<a href="3.html">Посилання №3</a></p>
<form action="">
<p><input type="text" value="Текстове поле №1"></p>
<p><input type="text" value="Текстове поле №2"></p>
</form>
<table border="1" width="100%" cellpadding="5">
<tr>
<th>Комірка №1 </th>
<th>Комірка №2 </th>
</tr>
<tr>
```

```
<td>Комірка №3 </td>
<td>Комірка №4 </td>
</tr>
</table>
 </body>
</html>
```

Існують структурні псевдокласи. До цієї групи належать псевдокласи, які визначають положення елемента в дереві документа, і застосовують до нього стиль залежно від його статусу.

:first-child – застосовується до першого дочірнього елемента селектора, який розташований в дереві елементів документа. Наприклад: *ul > li:first-child {margin-top: 10px;}*.

Аналогічно використовується псевдоклас *:last-child*.

Приклад 1.30

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Застосування псевдокласу :last-child </title>
<style type="text/css">
b:first-child {
color: blue; /* Червоний колір тексту */
}
p {
text-indent: 1em; /* Відступ першого рядка */
}
p:first-child {
text-indent: 0; /* Видаляємо відступ для першого абзацу */
}
</style>
</head>
<body>
<p><b>Lorem ipsum</b> dolor sit amet, <b>consectetuer</b>
adipiscing <b>elit</b>, sed diem nonummy nibh euismod tincidunt
ut lacreet dolore magna aliguam erat volutpat.</p>
<p><b>Ut wisis enim</b> ad minim veniam, <b>quis nostrud</b>
exerci tution ullamcorper suscipit lobortis nisl ut aliquip
ex ea <b>commodo consequat</b>.</p>
</body>
</html>
```

Псевдоелементи дають змогу задати стиль елементів, що не визначені в дереві елементів документа, а також генерувати вміст, якого немає у вихідному коді тексту.

Синтаксис використання псевдоелементів такий: *Селектор:Псевдоелемент { Опис правил стилю }.*

:first-letter – Задає стилі для першої літери всередині елемента (наприклад, перші літери абзаців сторінки мають інший стиль):

Приклад 1.31

```
p:first-letter {
font-size: 120%;
}
```

:first-line – Визначає стиль першого рядка блочного тексту.

:after :before – дають змогу вставити генерований контент на ходу після або до певного елемента. Щоб задати вставлений текст, використовується спеціальна властивість *content*. Властивість *content* задає вміст, який спочатку відсутній на сторінці. Виводиться до або після елемента, до якого цю властивість застосували. Використовується спільно з псевдоелементами *after* та *before*.

Приклад 1.32

```
<!DOCTYPE HTML>
<html>
```

```
<head>
<meta charset=windows-1251">
<title>Застосування псевдоелементів</title>
<style>
p:first-line {
color: red; /* Червоний колір тексту */
font-style: italic; /* курсив */
}
p:first-letter {
/* Гарнітура шрифту першої літери */
font-family: ‘Times New Roman’, Times, serif;
font-size: 200%; /* Розмір шрифту першого символу */
color: blue; /* Червоний колір тексту */
}
p:before {
/* Додаємо перед елементом списку символ в юнікод */
content: "\30aa ";
}
</style>
</head>
<body>
<p>Lorem ipsum dolor sit amet,consectetuer>
adipiscing elit, sed diem nonummy nibh euismod tincidunt
ut lacreet dolore magna aliguam erat volutpat.
Ut wisis enim ad minim veniam,quis nostrud
exerci tution ullamcorper suscipit lobortis nisl ut aliquip
ex ea commodo consequat.</p>
</body>
</html>
```

Використовуючи псевдокласи та псевдоелементи можна створювати динамічне меню навігації.

Меню дає змогу швидко переміщатися основними розділами сайта.

Меню може бути:

- Горизонтальним;
- Вертикальним;
- Багаторівневим.

Меню може бути дуже різноманітним, і зверстати його можна дуже різними способами: таблицями, блоками, просто посиланнями та ін.

Створимо горизонтальне меню за допомогою списків.

Приклад: 1.33

```
<!DOCTYPE HTML>
<html>
<head>
<title>Горизонтальне меню</title>
<style>
.menu_nav {
list-style: none; /* ховаємо маркери */
}
.menu_nav li {
float: left; /* розташовуємо елементи списку в один ряд */
margin-right: 15px; /* робимо відступ, щоб пункти меню не зливалися */
}
a {text-decoration: none;} /*Прибираємо підкреслення посилання*/
.menu_nav li a {
display: block; /* змінюємо відображення на блок, щоб мати можливість задавати внутрішні відступи */
padding: 3px 5px;
background: #ff7f50;
```

```
color: #000;
position: relative;
}
.menu_nav li a:hover {
background: #d2b48c;
color: #fff;
}
</style>
</head>
<body>
<ul class="menu_nav">
<li><a href="#">Головна</a></li>
<li><a href="#">Інститут</a></li>
<li><a href="#">Кафедри</a></li>
<li><a href="#">Розклад</a></li>
</ul>
</body></html>
```

Елемент * * – блоковий елемент який призначений для виділення фрагмента документа задля зміни виду даних. Зазвичай внаслідок вид блока управляється за допомогою стилів. Щоб не описувати щоразу стиль всередині елемента розмітки, можна виділити стиль у зовнішню таблицю стилів, а для елемента додати атрибут *class* або *id* з ім'ям селектора.

Як і при використанні інших блочних елементів, вміст елемента * * завжди починається з нового рядка. Після нього також додається перенесення рядка.

Завдяки цьому елементу HTML-код розпадається на ряд чітких наочних блоків, внаслідок чого верстка шарами називається також блокової верстkою. Код при цьому виходить більш компактним, ніж при табличній верстці, до того ж пошукові системи його краще індексують.

Розглянемо приклади сторінок (рис. 1.4).

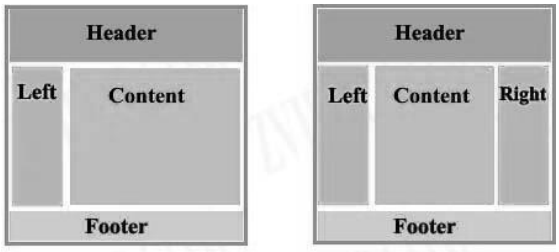

Рис. 1.4. Приклад для блочної розмітки

Тут наведено представлені варіанти для «гумової» верстки з двома і трьома колонками. Виконаємо розмітку двох колонок.

Спочатку виконаємо розмітку HTML:

Приклад 1.34

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset=UTF-8">
<title> Блокова верстка </title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<div id="main_container">
<div id="header">
<h1>Header</h1>
</div>
<div id="left">
<h3>left Column</h3>
</div>
<div id="content">
<h1>Main Content </h1>
```

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent aliquam, justo convallis luctus rutrum, erat nulla fermentum Integer turpis arcu, pellentesque eget, cursus et, fermentum ut, sapien. Fusce metus mi, eleifend sollicitudin, molestie id, varius et, nibh. Donec nec libero.</p>
<h1>Content</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>
<p>Quisque ornare risus quis ligula Phasellus tristique purus a augue condimentum adipiscing. Aenean sagittis. Etiam leo pede, rhoncus venenatis, tristique in, vulputate at, odio.</p>
</div>
<div id="footer">
<p><strong>Footer</strong></p>
</div>
</div>
</body>
</html>
```

Створення сторінки починається з головного контейнера в який додаються інші. Наявність цього контейнера дає змогу регулювати інші параметри сторінки, що значно полегшує верстку блоками.

Далі створюємо файл *style.css,* в який додаємо стилі.

Приклад 1.35

```
body, html {
margin:0px; /*Обнуляємо поля і відступи, тому що різні браузери по-різному їх сприймають. */
padding:0px;
text-align:center; /*Вирівнюємо макет (вміст елемента body буде посередині) по центру в старих версіях браузерів */
}
#container{
margin:0 auto; /*вирівнюємо макет по центру в сучасних браузерах */
width:650px;
}
/*Вводимо стилі для голови сайта */
#header{
background-color: #00cc66;
}
/* Вводимо для лівої колонки сайта */
#left{
background-color: #ccff33;
}
/* Вводимо стилі для блоку контента */
#content{
background-color: #0099ff;
}
#content h1 {
margin:0px; /* Обнуляємо відступи для заголовка першого рівня, що міститься в блоці контенту.*/
}
/* Вводимо стилі для підвалу сайта */
#footer{
background-color: #ffcc33;
}
```

Розміщуємо блок *left* ліворуч від блока Content. Змінюємо ширину блока *Left* на *150px* і встановлюємо для нього обтікання:

```
#left{
width:150px; /*ширина колонки */
float:left; /*обов'язкове вирівнювання по лівому краю,
з включенням обтікання*/
}
```

Зміщаємо блок Content праворуч на ширину блока *Left*:

```
#content {
```

```
background-color:#83a0f3;
margin-left:150px;
}
```

Те саме можна зробити інакше, встановивши блоку *Left width:20%* і *float:left;,* а *блоку Content width:80%* і *float:right.*

При цьому не забудьте поставити для блока *footer* значення *cltar:both*, інакше при збільшенні кількості тексту в попередніх блоках може вийти сивий ефект.

Завдання до самостійної роботи

1. Оформіть список дисциплін, що вивчаються на четвертому курсі за допомогою вивчених псевдокласів і псевдолелементов. Обов'язково використовуйте: псевдокласи посилань, динамічні псевдокласи, псевдоелементи.
2. Використовуючи *Приклади 1.34* та *1.35* розробіть трьостопчиковий блоковий макет. Як зразок візьміть двостопчиковий макет і розтягніть вміст на всю ширину сторінки. Використовуючи раніше створені сторінки – «Хобі» і «Автобіографія», створіть сайт оформлений в єдиному стилі.
3. Розробіть динамічне багаторівневе навігаційне меню для сайта з використовуючи вивчений матеріал.
4. З допомогою меню необхідно зв'яжіть усі наявні сторінки.
5. Сторінки повинні містити зображення.

Контрольні запитання та завдання

1. Назвіть призначення псевдокласів.
2. Які ви знаєте псевдоелементи?
3. Як за допомогою CSS виконати форматування тексту?
4. Яку структуру мають сторінки, створені за допомогою блокової верстки?