Тема: Зовнішній вигляд списків і блоків в CSS

**Активне застосування тегу **

При блоковій верстці істотне значення приділяється універсальному тегу , який виконує безліч функцій. Фактично це основа, на яку «навішуються» стилі. Звісно, це не означає, що застосовується тільки один цей тег, адже потрібно і малюнки вставляти і оформляти текст. Але при верстці за допомогою блоків тег є цеглинкою верстки, її базовим фундаментом.

Завдяки цьому тегу HTML-код розпадається на ряд чітких наочних блоків, код при цьому виходить більш компактним, ніж при табличній верстці, до того ж пошукові системи його краще індексують.

Таблиці застосовуються тільки для представлення табличних даних.

При блоковій верстці, звичайно ж, використовуються таблиці, але тільки в тих випадках, коли вони потрібні, наприклад, для наочного відображення чисел та інших табличних даних. Варіант, коли від таблиць пропонується відмовитися взагалі, є недоцільним і, більш того, шкідливим.

У HTML5 додано кілька нових тегів розмітки для позначення різних типових блоків сторінки. Наприклад, <header> і <footer> використовуються для створення «шапки» і «підвалу», <nav> для навігації, <aside> для бічної панелі. Увімкнення в специфікацію HTML подібних елементів покликане знизити домінування тегу і надання більшого сенсу розмітці. Тому у верстці на HTML5 активно застосовується термін «елемент», під яким мається на увазі відповідний тег і елемент, який він створює.

Викладені вище принципи блочної верстки при цьому зберігаються за винятком того моменту, що в деяких випадках замінюється більш осмисленими тегами.

Будь-який блоковий елемент складається з набору властивостей, що накладаються один на одного. Основою блоку виступає його контент (це може бути текст, зображення тощо), ширина якого задається властивістю width, а висота через height; навколо контенту йдуть внутрішні поля (padding), вони створюють порожній простір від контенту до внутрішнього краю рамки, потім йдуть власне сама рамка (border) і завершують блок зовнішні відступи (margin),

невидимий порожній простір від зовнішнього краю рамки. Порядок впливу цих властивостей на блок чітко визначений і не може бути порушений. На малюнку зображений блок у вигляді набору цих властивостей.

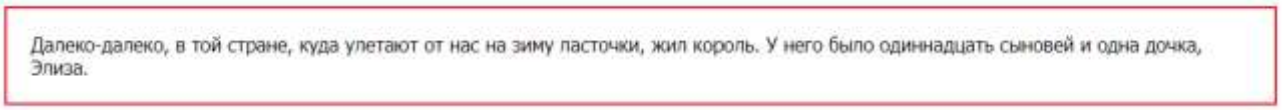
Поля

Полем будемо називати відстань від внутрішнього краю рамки або краю блоку до уявного прямокутника, що обмежує вміст блоку. Через те, що значення полів можуть відрізнятися на кожній стороні, застосовують вираз «верхнє поле» або «поле зверху», і їм подібні для інших сторін. Позначення «поля» слід розуміти, як однакове значення полів для всіх сторін. Основне призначення полів - створити порожній простір навколо вмісту блочного елемента, наприклад тексту, щоб він не прилягав щільно до краю елемента. Використання полів підвищує читабельність тексту і покращує зовнішній вигляд сторінки. У прикладі показано використання полів для оформлення тексту.

```
<style type="text/css">
  .space {
   padding: 20px; /* Поля */
   background: #E5D3BD; /* Цвет фона */
   border: 2px solid #E81E25; /* Параметры рамки */
  }
</style>
<div class="space">
```

Далеко-далеко, в той стране, куда улетают от нас на зиму ласточки, жил король.

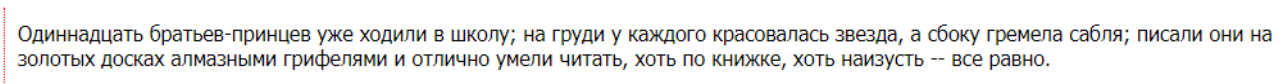
У него было одиннадцать сыновей и одна дочка, Элиза.



Рамки

Рамки це лінії навколо полів елемента на одній, двох, трьох або всіх чотирьох його сторонах. У кожної лінії є товщина, стиль і колір. Для створення рамки застосовується універсальна властивість border, що одночасно задає всі ці параметри, а для створення ліній на окремих сторонах елемента можна скористатися властивостями border-left, border-top, border-right і border-bottom, відповідно встановлюють кордон зліва, зверху, праворуч і знизу . У прикладі показано додавання лінії зліва від елементу:

```
style type="text/css">
  .line {
   border-left: 1px dotted red;
   padding: 10px;
  }
</style>
<div class="line">
  Одиннадцать братьев-принцев уже ходили в школу;
  на груди у каждого красовалась звезда, а сбоку гремела сабля;
  писали они на золотых досках алмазными грифелями и отлично
  умели читать, хоть по книжке, хоть наизусть -- все равно.
</div>
```



Відступи

Відступом будемо називати порожній простір зовні від зовнішнього краю рамки, полів або вмісту блоку. Рамки та поля не є обов'язковими і можуть бути

відсутні, так що спосіб формування відступів залежав від ситуації. Як і у випадку з полями, застосовують вираз «верхній відступ» або «відступ зверху», і їм подібні для інших сторін. Позначення «відступи» слід розуміти як однакове значення відступів для всіх сторін.

Для відступів характерні наступні особливості.

- Відступи прозорі, на них не поширюється колір фону або фонова картинка, визначена для блоку. Однак, якщо фон встановлений у батьківського елементу, він буде помітний і на відступи.
- Відступи на відміну від полів можуть приймати від'ємне значення, це призводить до зсуву всього блоку у вказаний бік. Так, якщо задано margin-left:-10px, це зрушить блок на десять пікселів вліво.
- Для відступів характерне явище, коли відступи у прилеглих елементів не підсумовуються, а об'єднуються між собою.
- Відступи, задані у відсотках обчислюються від ширини блоку. Це стосується як вертикальних, так і горизонтальних відступів.

У прикладі показано відступи і їх прозорість.

```
<style type="text/css">
  .layer1, .layer2 {
   background: #F2EFE6;
   border: 1px solid #B25538;
   padding: 10px;
   margin: 20px;
  }
</style>
<div class="layer1">
  Отец их, король той страны, женился на злой королеве,
  которая невзлюбила бедных детей. Им пришлось испытать
  это в первый же день: во дворце шло веселье, и дети затеяли
  игру в гости, но мачеха вместо разных пирожных и печеных яблок,
  которых они всегда получали вдоволь, дала им чайную чашку
  песку и сказала, что они могут представить себе, будто это угощение.
```

```
</div>
<div class="layer2">
  Через неделю она отдала сестрицу Элизу на воспитание в деревню
  каким-то крестьянам, а прошло еще немного времени, и она успела
  столько наговорить королю о бедных принцах, что он больше
  и видеть их не хотел.
</div>
```

Ширина блоку

Ширина блоку це комплексна величина і складається з декількох значень властивостей:

- Width - ширина контенту, тобто вмісту блоку;
- Padding-left і padding-right - поле ліворуч і праворуч від контенту;
- Border-left і border-right - товщина кордону ліворуч і праворуч;
- Margin-left і margin-right - відступ зліва і справа.

Якісь властивості можуть бути відсутні і в цьому випадку на ширину не впливають. Загальна ширина показана на малюнку у вигляді чорної пунктирної лінії.

Якщо значення width не задано, то воно за умовчанням набуває значення auto. У цьому випадку ширина блоку буде займати всю доступну ширину при збереженні значень полів, меж і відступів. Під доступною шириною в даному випадку мається на увазі ширина контенту у батьківському блоці, а якщо його немає, то ширина контенту браузера.

Припустимо, для блоку написаний наступний стиль.

width: 300px; / * Ширина * /

margin: 7px; / * Відступи * /

border: 4px solid black; / * Параметри рамки * /

padding: 10px; / * Поля навколо тексту * /

Ширина блоку згідно цього запису дорівнюватиме 342 пікселя, ця величина виходить складанням значення ширини контенту плюс відступ зліва, рамка зліва і поле ліворуч, плюс поле праворуч, рамка справа і відступ справа. Додаємо всі числа:

Ширина = 300 + 7 + 7 + 4 + 4 + 10 + 10 = 342

Треба зазначити, що блокова модель з формуванням ширини несе в собі купу незручностей. Постійно доводиться займатися обчисленнями, коли потрібно задати певну ширину блоку. Також починаються проблеми при поєднанні різних одиниць виміру, зокрема, відсотків і пікселів. Припустимо, що

ширина контенту задана як 90%, якщо сюди приплюсувати поля і межі, задані в пікселях, то не можна обчислити сумарну ширину блоку, оскільки відсотки безпосередньо пікселі не переводяться (в CSS3 так підсумовувати можна, тільки підтримується ця можливість далеко не всіма браузерами). У результаті може вийти так, що загальна ширина блоку перевищить ширину веб-сторінки, що призведе до появи горизонтальної полоси прокрутки. Виходів із такої ситуації два - поміняти алгоритм блокової моделі і скористатися вкладеними блоками.

Вкладені блоки

Ідея проста - для зовнішнього блочного елемента задається тільки необхідна ширина, а для вкладеного блоку все інше - поля, межі та відступи. Оскільки за замовчуванням ширина блоку дорівнює доступною батьківській ширині, то вийде, що блоки в якомусь сенсі накладаються один на одного, при цьому фактична ширина такого комбінованого елемента буде чітко задана. У прикладі показано використання вкладених блоків.

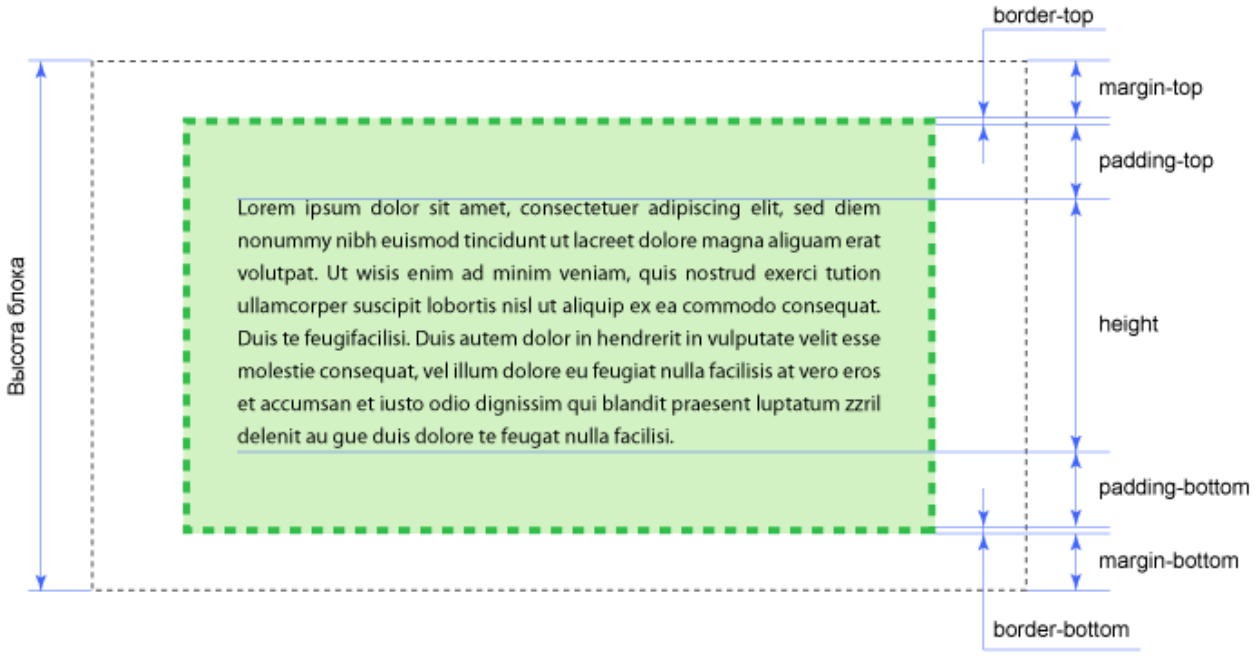
```
<style type="text/css">
  span.inlineBlock{
     display:inline-block;
     vertical-align:top;
  }
.parent{
  width:100%;
  background-color:red;
}
.child1{
  width:40%;
  background-color:yellow;
}
.child2{
  width:60%;
  background-color:green;
}
```

```
.blk {
   padding: 20px;
   border: 2px solid #E81E25;
   margin: 10px;
}
</style>
<span class="inlineBlock parent">
 <span class="inlineBlock child1"><div class="blk">
  -- Летите-ка подобру-поздорову на все четыре стороны! -- сказала злая королева. -- Летите большими птицами без голоса и промышляйте о себе сами!
 </div></span><span class="inlineBlock child2"><div class="blk">
  Но она не могла сделать им такого зла, как бы ей хотелось, -- они превратились в одиннадцать прекрасных диких лебедей, с криком вылетели из дворцовых окон и понеслись над парками и лесами. Было раннее утро, когда они пролетали мимо избы, где спала еще крепким сном их сестрица Элиза. Они принялись летать над крышей, вытягивали свои гибкие шеи и хлопали крыльями, но никто не слышал и не видел их; таким пришлось улететь ни с чем. Высоко-высоко взвились они к самым облакам и полетели в большой темный лес, что тянулся до самого моря.
 </div></span>
</span>
```

Висота блоку

На висоту блоку діють ті ж правила, що і на ширину. А саме, висота складається із значень висоти контенту (height), полів (padding), рамок (border) і відступів (margin). Якщо властивість height не зазначено, то вона вважається як auto, в цьому випадку висота контенту обчислюється автоматично на основі

вмісту. На малюнку показані властивості, що дають підсумкову висоту, яка позначена чорною пунктирною лінією.



Разом з тим, незважаючи на схожість принципів побудови ширини і висоти, у них є суттєві відмінності. Це стосується того випадку, коли значення width і height не вказано, тоді за замовчуванням воно приймається як auto. Для ширини блоку - це максимально доступна ширина контенту, а для висоти блоку - це висота контенту. Також для ширини блоку відома батьківська ширина, навіть якщо вона не вказана явно. Це дозволяє встановлювати значення width у відсотках. Використання ж відсотків для height ні до чого не призведе, тому що батьківська висота не обчислюється і її треба вказувати. У прикладі показано, як задати висоту блоку у %.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Высота блока</title>
  <style type="text/css">
   html, body {
    height: 100%; /* Висота зовнішнього елемента */
```

```
 margin: 0; /* прибираємо відступи */
}
div {
 height: 100%; /* Висота */
 background: #fc0;
 margin: 10px;
 padding: 20px;
 border: 1px solid #000;
}
</style>
</head>
<body>
 <div>Висота  100%</div>
</body>
</html>
```

Для тегу в прикладі батьківським виступає тег <body>, тому для нього встановлюємо значення height рівним 100%. У той же час на <body> діють ті ж правила, що і на , тому величина у відсотках обчислюватиметься не від висоти сторінки, а від висоти контенту. Так що для батьківського <body>, яким є тег <html>, також потрібно поставити значення height рівним 100%. Тільки в цьому випадку висота блоку у відсотках буде залежати від висоти сторінки.

Оскільки на висоту впливає значення полів, меж і відступів, у прикладі з'явиться вертикальна смуга прокрутки. Позбутися від цього впливу можна тими ж методами, що і для ширини, а саме, використовувати вкладені блоки.

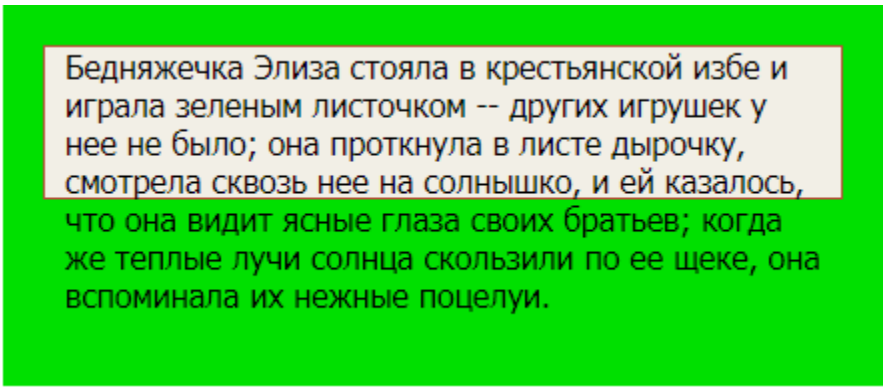
З висотою пов'язана ще одна особливість - при перевищенні вмісту блоку його розмірів при заданій висоті, вміст починається відображатися поверх блоку.

```
<style type="text/css">
  div.textoverflow {
   height: 75px;
   background: #F2EFE6;
   border: 1px solid #B25538;
```

```
 padding: 0 10px;
}
</style>
<div class="textoverflow">
```

Бедняжечка Элиза стояла в крестьянской избе и играла зеленым листочком -- других игрушек у нее не было; она проткнула в листе дырочку, смотрела сквозь нее на солнышко, и ей казалось, что она видит ясные глаза своих братьев; когда же теплые лучи солнца скользили по ее щеке, она вспоминала их нежные поцелуи.

```
</div>
```



Щоб уникнути подібних неприємностей, висоту контенту краще не ставити, тоді висота блоку буде обчислюватися автоматично. Втім, бувають випадки, коли висота має бути чітко зазначена, тоді рекомендується до стилю додати властивість overflow із значенням auto або hidden. Результат у них різний, auto додає смуги прокрутки автоматично, коли вони потрібні, hidden приховує все, що не поміщається в задані розміри.

Приклад Зовнішній вигляд блоків

Приклад. Опрацювати і проаналізувати даний приклад, який складається з двох файлів: **index.html і main.css**

ФАЙЛ : index.html

```
<html>
<head>
  <title>Вивчаємо CSS</title>
  <link type="text/css" rel="stylesheet" href="css/main.css" />
</head>
```

```
<body>
  <div id="container">
    <p>Текст всередині блоку.</p>
    <p>Текст всередині блоку.</p>
    <a href="#">Посилання</a>
    <p>Текст всередині блоку.</p>
  </div>
  <p>Текст за межами div.</p>
  <div class="div_1">
    <div class="div_2"></div>
  </div>
  <div class="over">Тут багато тексту ... Тут багато тексту ... Тут багато тексту ... Тут багато тексту ... Тут багато тексту ...</div>
  <div class="over">Тут багато тексту ...</div>
</body>
</html>
```

ФАЙЛ : main.css

```
#container {
  border: 2px solid #000;
  margin: 20px auto;
  width: 50%;
}

p {
  border: 1px solid #c00;
  margin: 0;
}

a {
```

```
    border: 1px solid #0c0;
}

p:first-child {
    /*padding: 5px;*/
    padding-bottom: 9px;
    padding-left: 2px;
    padding-right: 4px;
    padding-top: 5px;
    margin: 10px;
}

p:last-child {
    padding: 5px 10px 20px 30px;
    margin-bottom: 25px;
    margin-left: 55px;
    margin-right: 15px;
    margin-top: 15px;
    margin: 10px 15px 20px 30px;
}

body > p {
    border: 2px solid #c00;
    margin: 20px 30px; /* = margin: 20px 30px 20px 30px; */
    margin: 20px 30px 0; /* = margin: 20px 30px 0 30px; */
}

.div_1 {
```

```
    border: 1px solid #000;
    border-radius: 25px;
    box-shadow: 5px 4px 10px 15px #0c0;
    height: 400px;
    width: 500px;
}

.div_2 {
    border-bottom: 1px dashed #00c;
    border-left: 1px dotted #c00;
    border-top: 1px solid #cc0;
    border-right: 1px solid #333;
    height: 200px;
    margin: 0 auto;
    width: 70%;
}

.over {
    border: 1px solid #000;
    font-size: 150%;
    height: 80px;
    margin: 20px 0;
    overflow: auto;
    width: 200px;
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

Приклад Зовнішній вигляд списків

Приклад. Опрацювати і проаналізувати даний приклад, який складається з двох файлів: **index.html і main.css** папки **images** в якій знаходиться файл 

ФАЙЛ : index.html

```
<!DOCTYPE html>
<html>
<head>
        <title>Вивчаємо CSS</title>
   <link type="text/css" rel="stylesheet" href="css/main.css" />
</head>
<body>
   <ul>
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
   </ul>
   <ul class="icon">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
   </ul>
   <h2>Меню</h2>
   <nav>
      <ul>
         <li>
            <a href="#">Головна</a>
         </li>
```

```
            <li>
                <a href="#">Профіль</a>
            </li>
            <li>
                <a href="#">Вихід</a>
            </li>
        </ul>
    </nav>
</body>
</html>
```

ФАЙЛ : main.css

```
body {
    font-size: 30px;
}

ul {
    list-style-type: none;
}

ul.icon {
    list-style-image: url("../images/icon.png");
}

nav ul {
    margin: 0;
    padding: 0;
}

nav li {
    line-height: 130%;
}
```

```
nav a {
   text-decoration: none;
}

nav a:hover {
   color: #c00;
}
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

ЗАВДАННЯ

Завдання 1.

Виведіть за допомогою ul список Ваших справ на завтра.

Завдання 2.

Замініть маркер за замовчуванням на якусь невелику картинку.

Завдання 3.

Виведіть блок header і footer, додавши туди текст.

Завдання 4.

Додайте відступи і поля в header і footer на свій розсуд, а також вирівняти їх вміст по центру.

Завдання 5.

В середині header додайте div, а після задайте у нього ширину, висоту, рамку і тінь на свій розсуд.

ТЕМА Завдання фону. Градієнт в CSS.

Кольорове виділення інформації та фон, на якому вона розміщена, мабуть, перше, що кидається в очі при завантаженні веб - сторінки, тому з них ми і почнемо знайомство з *CSS* .

За керування кольором і фоном в *CSS* відповідають такі атрибути стилів, підтримувані абсолютною більшістю елементів:

- *color* - задає колір переднього плану (*color* : # 00FF00);
- *background-color* – задає *колір фону* елемента (*background-color* : brown);
- *background-image* - задає фонове зображення для елемента (*background-image* : url ("*image .gif* "));
- *background-repeat* - задає тип повторення зображення, встановленого за допомогою атрибута стилю *background-image* (*background-repeat* : no-*repeat*) , може набувати таких значень:
    - repeat -x - зображення повторюється по горизонталі;
    - repeat -y - зображення повторюється по вертикалі;
    - repeat - зображення повторюється по горизонталі і вертикалі;
    - no-repeat - зображення не повторюється (значення по - замовчуванню).
- *background-attachment* - визначає чи буде фонове зображення прокручуватися разом з елементом (*background-attachment* : fixed), може набувати таких значень:
    - scroll - зображення буде прокручуватися разом з елементом;
    - fixed - прокрутка зображення заблокована.
- *background-position* - визначення координат позиціонування фонового зображення, містить два значення: положення по горизонталі і положення по вертикалі (*background-position* : 5cm 4cm). Крім числових, може набувати таких значень:
    - left – горизонтальне позиціонування "по лівому краю";
    - center – горизонтальне позиціонування "по центру";
    - right – горизонтальне позиціонування "по правому краю";
    - top – вертикальне позиціонування "зверху";

- center – вертикальне позиціонування "по центру";
- bottom – вертикальне позиціонування "знизу".

Можна задати всі атрибути стилю, що відносяться до фонового зображення, скориставшись короткою формою запису, наприклад:

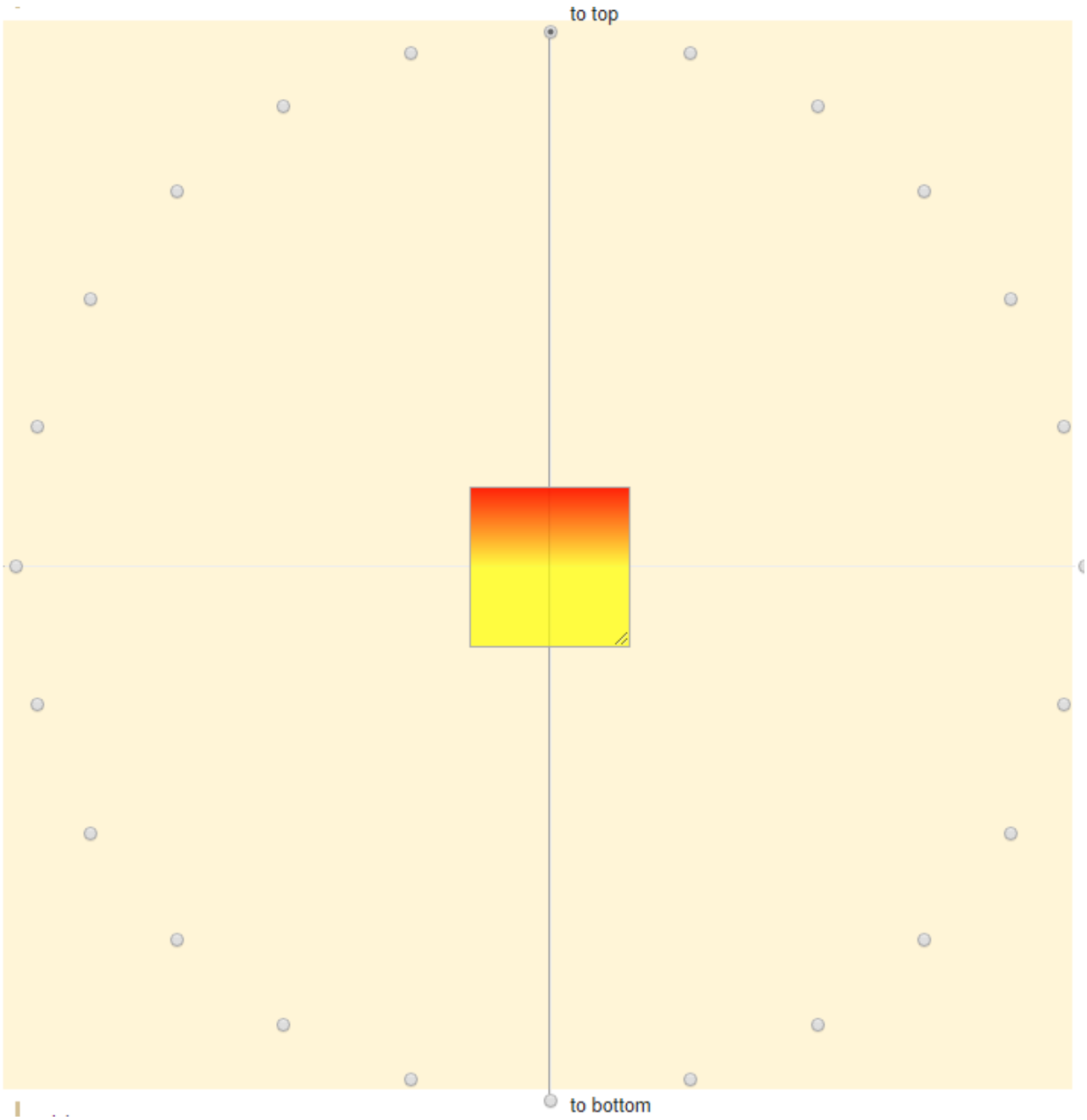
background: # 00FF00 url ("image.gif") no-repeat fixed 5cm 4cm

ГРАДІЄНТ

Лінійний градієнт linear-gradient. Лінійний градієнт поширюється (частіше зверху вниз) по лінії складається з двох і більше відтінків

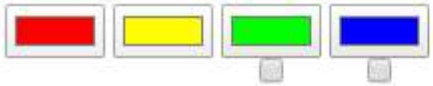
```
<style>
.div {
  background-image: linear-gradient(#ff0000, #ffff00);
  width: 130px;
  height: 130px;
  border: 1px solid #aaa;
}
</style>

<div class="div"></div>
```

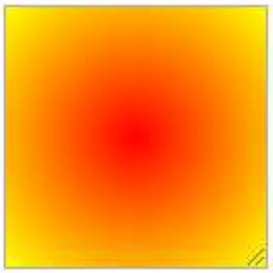
можна змінювати напрямок градієнта, якщо задати градус кута **deg** або задати після **to** ключові слова [Left | right] || [Top | bottom]




```
<style>
.div {
  background-image: linear-gradient(180deg, rgba(255,0,0,1) -10px, rgba(255,255,0,.7) 50%);
  /* или */ background-image: linear-gradient(to top, rgba(255,0,0,1) -10px, rgba(255,255,0,.7) 50%);
  width: 130px;
  height: 130px;
  border: 1px solid #aaa;
}
</style>
```

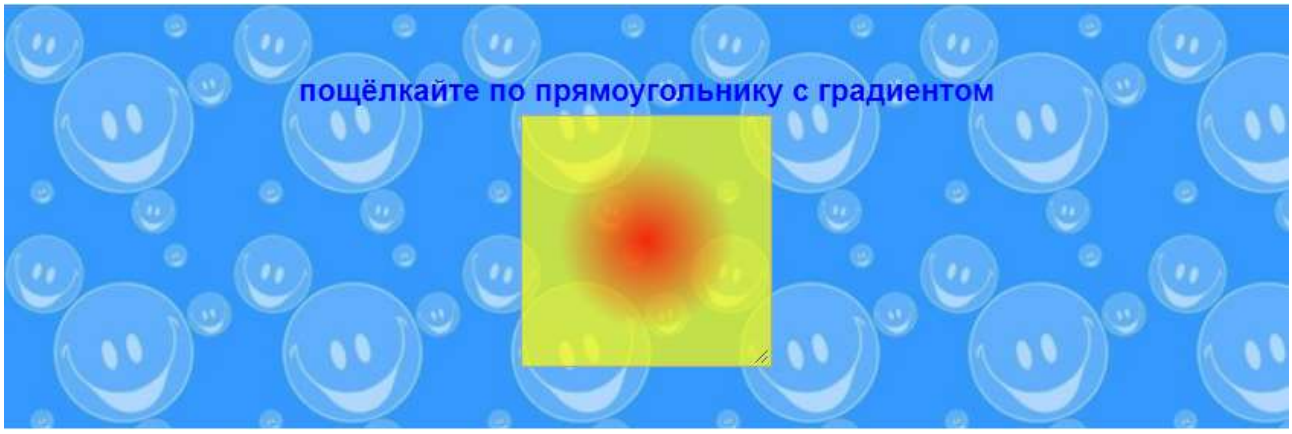
Радіальний градієнт radial-gradient радіальний градієнт поширюється по окружності складається з двох і більше відтінків





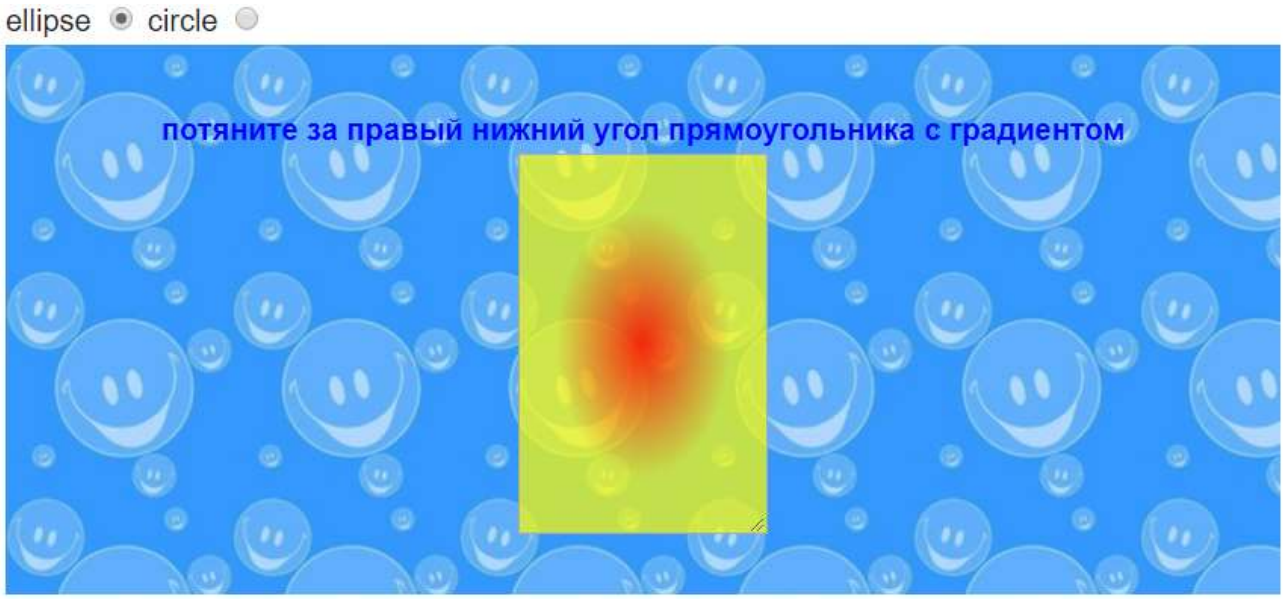
```
<style>
.div {
  background-image: radial-gradient(#ff0000, #ffff00);
  width: 130px;
  height: 130px;
  border: 1px solid #aaa;
}
</style>

<div class="div"></div>
```

За замовчуванням точкою відліку є середина **at center center.** Її можна зміщувати аналогічно **background-position**. Перше значення визначає розташування по горизонталі, друге - вертикалі. Вказуються після прийменника **at** в будь-якій одиниці виміру, у відсотках або при використанні ключових слів [Left | center | right] || [Top | center | bottom]




```
<style>
.div {
  background-image: radial-gradient(at center center, rgba(255,0,0,1) -10px, rgba(255,255,0,.7) 50%);
  width: 130px;
  height: 130px;
  border: 1px solid #aaa;
}
</style>

<div class="div"></div>
```

За замовчуванням окружність має вигляд еліпса, але її можна змінити на коло.




```
<style>
.div {
```

```
 background-image: ;
 width: 130px;
 height: 130px;
 border: 1px solid #aaa;
}
</style>

<div class="div"></div>
```

Синтаксис радіального градієнта:

background-image: -XXX-radial-gradient (позиція, форма розмір, колір%, колір%);

Приклад Задання фону

Приклад. Опрацювати і проаналізувати даний приклад, який складається з двох файлів: **index.html і main.css**

ФАЙЛ : index.html

```
<html>
<head>
   <title>Вивчаємо CSS</title>
   <link type="text/css" rel="stylesheet" href="css/main.css" />
</head>
<body>
   <div class="red"></div>
   <div class="image_blue"></div>
   <div class="image_blue"></div>
   <div class="bg_size"></div>
   <div class="bg_mult"></div>
</body>
</html>
```

ФАЙЛ : main.css

```
body {
   background: url("../images/bg.jpg") repeat;
```

```
    margin: 0;
    padding: 0;
}

.red {
    background-color: #c00;
    margin: 40px;
    height: 400px;
    padding: 40px;
    width: 400px;
}

.red:hover {
    background-image: url("../images/bg.jpg");
    background-repeat: no-repeat;
}

.image_blue {
    border: 10px solid #000;
    background: url("../images/bg.jpg") no-repeat 20.4% 10% #00c;
    height: 400px;
    width: 400px;
}

.image_blue:last-child {
    background-position: bottom center;
}

.bg_size {
    border: 10px solid #c00;
    background: url("../images/bg.jpg") no-repeat #ccc;
```

```
    background-size: cover;
    height: 500px;
    width: 500px;
}

.bg_mult {
    background: url("../images/bg.jpg") no-repeat top left, url("../images/icon.png") repeat bottom right;
    border: 10px solid #cc0;
    height: 700px;
    width: 900px;
}
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

Приклад Градієнт

Приклад. Опрацювати і проаналізувати даний приклад, який складається з двох файлів: index.html і main.css

ФАЙЛ : index.html

```
<html>
<head>
    <title>Вивчаємо CSS</title>
    <link type="text/css" rel="stylesheet" href="css/main.css" />
</head>
<body>
    <div class="lg"></div>
    <div class="lg_2"></div>
    <div class="rg"></div>
</body>
</html>
```

ФАЙЛ : main.css

```
div {
```

```
    height: 500px;
    width: 500px;
}

.lg {
    background: linear-gradient(180deg, #c00, #000);
}

.lg_2 {
    background: linear-gradient(45deg, #c00, #0c0 30%, #000);
}

.rg {
    background: radial-gradient(at center center, #000, #c00);
    border-radius: 300px;
}
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

ЗАВДАННЯ

Завдання 1.

Знайдіть якесь не дуже велике зображення у себе на комп'ютері або в Інтернеті.

Завдання 2.

Зробіть фон у вигляді безлічі цих зображень у body, також прибравши margin і padding у нього.

Завдання 3.

Виведіть великим розміром тексту заголовок на сторінці і кілька абзаців тексту під ним.

Завдання 4.

Зробіть лінійний градієнтний фон у заголовка і кругової фон у всіх абзаців.

ТЕМА: Обтікання і позиціонування блоків в CSS.

Перш ніж розглядати *позиціонування* , в двох словах нагадаємо, що з точки зору боксової моделі кожен елемент *HTML* являє собою *прямокутник* (бокс), для якого можна задати такі параметри як поля, межі та заповнення. *Позиціонування* визначає, де повинен розташовуватися цей *прямокутник* , а також як він повинен впливати на елементи навколо себе. За допомогою позиціонування можна розмістити будь-який елемент точно в потрібному місці сторінки. Разом зі спливаючими елементами, розглянутими в "Модель компонування CSS" ,*позиціонування* дає великі можливості для створення оригінального дизайну.

В основі позиціонування лежить *уявлення* вікна браузера як системи координат. Будь-який бокс можна розмістити в цій системі координат де завгодно. Наприклад, наведене нижче правило дозволяє розташувати заголовок на відстані 100 пікселів від верхньої межі документа і на 200 пікселів від лівої межі документа:

```
H1 {
position: absolute;
top: 100px;
left: 200px;
}
```

Результат виконання даного коду представлений на мал.1

Мал.1 Приклад позиціонування заголовка

З наведеного прикладу зрозуміло, що для позиціонування елементів використовується властивість **position** . Властивість **position** має чотири

значення **static , relative , absolute і fixed** , які визначають тип позиціонування і впливають на розташування елемента.

Статичне позиціонування

Значення **static** властивості **position** використовується за замовчуванням. Будь-який елемент зі статичним позиціонуванням знаходиться в загальному потоці документа. Правила для його розміщення визначаються боксовою моделлю, блокові і *рядкові елементи* розміщуються за різними правилами, чітко визначеними в специфікації*CSS2*.1.

За замовчуванням, блокові бокси викладаються вертикально зверху вниз в порядку появи їх у розмітці. Кожен бокс зазвичай займає всю ширину документа і має розрив рядка перед і після себе. Вертикальна відстань між двома блоковими боксами управляється властивістю *margin-bottom* першого блоку і властивістю *margin-top* другого боксу, причому *вертикальні поля* між двома послідовними блоковими боксами будуть перекриватися таким чином, що відстань між ними буде визначатися не сумою двох полів, а великим з них.

Строкові бокси шикуються по горизонталі в тому порядку, в якому вони з'являються в розмітці, переходячи на новий рядок, тільки якщо вичерпано доступний горизонтальний простір. Залежно від властивості direction, строкові бокси будуть розташовуватися або зліва направо (direction:*ltr*), або справа наліво (direction:*rtl*).

Безліч строкових боксів, які складають рядок на екрані, зосереджуються ще в одному прямокутнику, тобто лінійному боксі. Лінійні бокси викладаються вертикально в своїх предків блочного рівня без додаткових прогалин між ними. Висотою лінійних боксів можна керувати за допомогою властивості line-height . Для строкових боксів не можна визначити розміри і *вертикальні поля* .

Відносне позиціонування

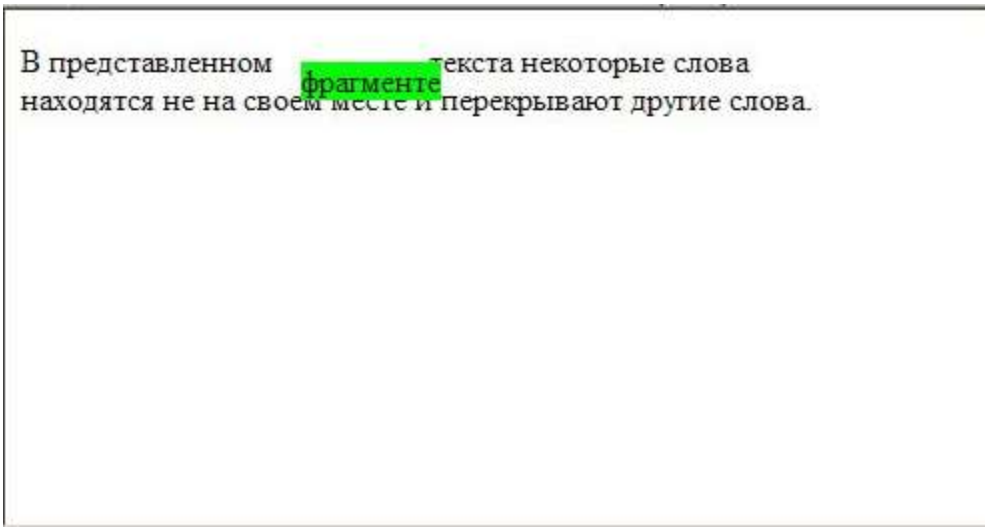
Елемент із значенням властивості **position** , рівним **relative** , спочатку розміщується за правилами статичного позиціонування. Але потім згенерований бокс зміщується щодо свого положення в потоці, відповідно до значень властивостей **top , bottom , left і right,** при цьому з потоку він не виключається, а продовжує займати там своє місце. Тобто зсувається зі свого місця він тільки

візуально, а положення всіх боксів навколо нього ніяк не змінюється. Це означає, що зміщений бокс може перекривати бокси інших елементів, оскільки вони як і раніше діють, тобто елемент, що відносно позиціонується, залишився там, де він мав бути перед застосуванням позиціонування. Як приклад відносного позиціонування, спробуємо змістити текст, поміщений в елемент <SPAN>.</SPAN> відносно його початкового положення на сторінці за допомогою наступного фрагмента коду :

```
SPAN {
  position: relative;
  top: 10px;
  left: 10px;
  background-color: lime;
}
...
```

<P> В представленому <SPAN> фрагменті </ SPAN> тексту деякі слова знаходяться не на своєму місці і перекривають інші слова. </ P>

Результат виконання даного коду представлений на малюнку.2 . Блок тепер перекриває наступний рядок тексту, а на його колишньому місці з’явився порожній простір.



Мал.2. Приклад щодо позиціонування.

Абсолютне і фіксоване позиціонування

Бокс з абсолютним позиціонуванням розташовується за заданими координатами, а з того місця, де він повинен би бути, він видаляється, і в цьому

місці відразу починають розміщуватися наступні бокси. Вважається, що бокс виключається з потоку.

Для *абсолютного позиціонування елемента* властивість **position** має приймати значення **absolute** . А для завдання положення розміщення блоку використовуються значення **left , right , top і bottom .** Всі чотири властивості можна використовувати одночасно для визначення відстані від кожного краю позиціонованого елементу до відповідного краю браузера або батьківського блоку. Можна визначити також позицію одного з кутів абсолютно позиціонованого (наприклад, використовуючи top і left), а потім визначити розміри боксу, використовуючи width і height.

Представлений нижче код дозволяє розмістити чотири бокси в різних кутах HTML-документа:

```
div # yellow1 {
  position: absolute;
  top: 10px;
  left: 10px;
}

div # yellow2 {
  position: absolute;
  top: 50px;
  right: 50px;
}

div # yellow3 {
  position: absolute;
  bottom: 10px;
  right: 10px;
}

div # yellow4 {
```

```
  position: absolute;
  bottom: 10px;
  left: 10px;
}
```

Результат застосування описаних вище властивостей представлений на малюнку 3 .

Мал.3. Приклад абсолютного позиціонування

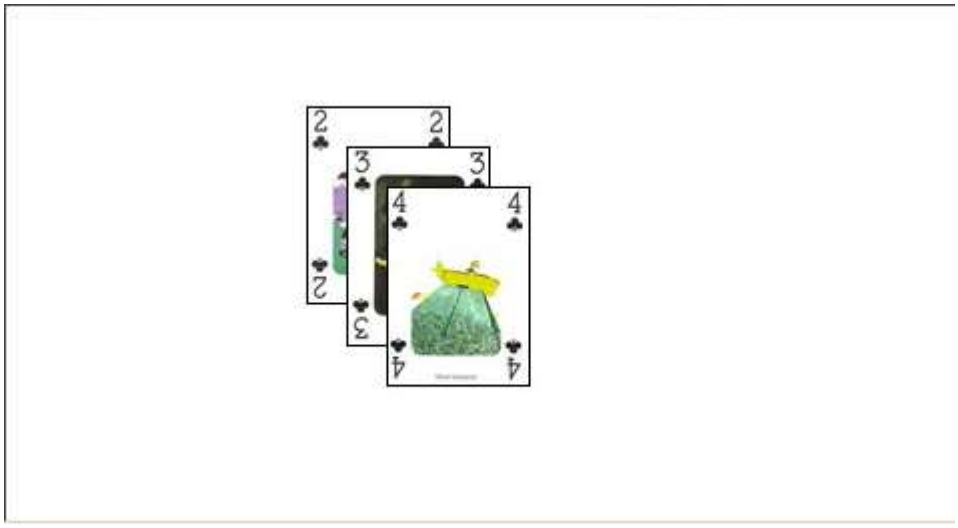
Фіксоване позиціювання діє подібно абсолютному, проте елемент з фіксованим позиціонуванням завжди розташовується тільки щодо вікна браузера і ніколи не зміщується при прокручуванні веб-сторінки.

Третій вимір веб-сторінки

Сторінка сайту двовимірна: для неї задані ширина і висота. CSS дозволяє додати веб-сторінці глибину (третій вимір) за допомогою властивості **z-index** . Дана властивість дозволяє створювати шари і розташовувати одні елементи поверх інших.

Для створення шарів необхідно для кожного елемента задати значення властивості **z-index** , яке є своєрідним порядковим номером шару, в якому знаходиться даний елемент. Це значення може бути цілим числом (яке може бути негативним) або одним з ключових слів **auto** або **inherit** . Заводський параметр **auto** або **0** . Елемент з великим значенням **властивості z-index** перекриває елемент з меншим значенням цієї ознаки. Нижче представлений код, що розташовує гральні карти в порядку зростання Результат виконання даного коду представлений на малюнку 4 .

```
div # card1 {
   position: absolute;
   top: 50px;
   left: 150px;
   z-index: 1;
}

div # card2 {
   position: absolute;
   top: 70px;
   left: 170px;
   z-index: 2;
}

div # card3 {
   position: absolute;
   top: 90px;
   left: 190px;
   z-index: 3;
}
```



Мал. 4. Приклад використання властивості z-index

Визначення float

Давайте почнемо з визначення, що таке **float** .

Float це умовно коробка, яка рухається вправо або вліво по поточній лінії. Найцікавіша характеристика ***float*** *в тому, що контент може обтікати уздовж його боку. При застосуванні властивості* ***float: left,*** *контент буде обтікати коробку вниз з правого боку і аналогічно при* ***float: right*** *- вниз з лівого боку.*

Властивість **float** має 4 значення, які ми можемо застосовувати: **left, right, inherinti none**. Кожне значення досить зрозуміло. Наприклад, якщо ви використовуєте **float: left** до елементу, то він переміститься в крайню ліворуч межу відносно свого батьківського елементу. І, якщо ви використаєте **float: right**, то елемент аналогічно переміститься в право. Значення **ihnerit** вказує елементу успадкувати властивість від свого батьківського елемента. І останнє значення **none** є значенням за замовчуванням і вказує не застосовувати властивість **float** до даного елементу.

Ось простий **Приклад 5** обтікання тексту навколо зображення і нижче відповідний CSS:

```
img                                                    {
  float:right;
  margin-top:10px;
}
```

Поведінки floats

Проаналізуємо світ float-ів і вияснимо, що насправді відбувається. У світі web, наш HTML пов'язаний декількома правилами, зокрема, правило нормального потоку. При нормальному потоці, кожен блоковий елемент (**div, p, h1, etc**) розташовується вертикально один над одним, починаючи з верхньої частини вікна і далі вниз. Елементи з властивістю **float** вилучаються з нормального потоку і відправляються в крайній правий або лівий край свого батьківського елементу. Іншими словами, вони перестають розташовуватися один над одним і стають один біля одного, підкреслюючи тим, що в батьківському елементі достатньо місця, щоб розташувати поруч всіх елементів з властивістю **float** . Важливо пам'ятати дану поведінку при розробці вашого сайту.

Давайте розглянемо ще один Приклад 6, в ньому три блоки без застосування властивості **float** .

```
.block{
width: 100px;
    height: 100px;
  }
```

Зверніть увагу що всі блоки розташовані вертикально один над одним. Це і є концепція нормального потоку. Нижче черговий Приклад 7 , тільки цього разу до елементів ми застосуємо властивість **float: left**.

```
.block{
float:left;
width: 100px;
height: 100px;
  }
```

In the clear

Властивість **float** має подібну властивість до **clear**. Вони доповнюють одна одну. Елемент із властивістю **float** перший, що вилучається з нормального потоку. Це означає, що кожен елемент, який буде йти за float-ed елементом, буде вести себе по іншому. У прикладі 8 задається властивість float двом блокам (рожевий і синій) і відразу після них додамо ще два блоки (зелений і помаранчевий) без **float**. Нижче код до даного приклад 8 :

```
<div class="block pink float"></div>
<div class="block blue float"></div>
<div class="block green"></div>
<div class="block orange"></div>
```

```
.block {
   width: 200px;
   height: 200px;
}
.float { float: left; }
.pink { background: #ee3e64; }
```

.blue { background: #44accf; }

.green { background: #b7d84b; }

.orange { background: #E2A741; }

Встановлюючи значення **clear: left** до нашого зеленого блоку, ми вказуємо йому вести себе, як ніби рожевий блок, що знаходиться в нормальному потоці, незважаючи на те що він був вилучений, і розташується під ним. Це дуже потужна властивість, вона допомагає повернути наші елементи без **float** властивостей в нормальний потік, тобто до поведінки, яку ми очікували за замовчуванням.

Властивість **clear** може отримувати 6 значень:

None Без завдання. Дозволяє обтікання елемента з усіх сторін.

Left Скасовує обтікання з лівого краю елемента. При цьому всі елементи, що розташовані ліворуч будуть опущені вниз, тобто, розташуються під поточним елементом.

Right Скасовує обтікання з правого боку елемента. При цьому всі інші елементи розташовані праворуч будуть опущені вниз і розташуються під поточним елементом.

Both Скасовує обтікання елемента одночасно з правого і лівого краю.

Initial Встановлює властивість у значення без завдання

Inherit Вказує на спадковість властивості від свого батьківського елемента

Приклад обтікання блоків

Приклад. Опрацювати і проаналізувати даний приклад, який складається з двох файлів: index.html і main.css та папки **images** в якій знаходиться файл 

ФАЙЛ : index.html

```
<html>
<head>
   <title>Вивчаємо CSS</title>
   <link type="text/css" rel="stylesheet" href="css/main.css" />
</head>
<body>
   <div>
      <img src="images/bg.jpg" alt="" />
      <p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
            <p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
```

```
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки.</p>
<p>Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки. Опис цієї картинки.  Опис цієї картинки. </p>
</div>
<nav>
  <ul id="topmenu">
    <li>
      <a href="#">Головна</a>
    </li>
    <li>
      <a href="#">Про нас</a>
    </li>
```

```
            <li>
                <a href="#">Контакти</a>
            </li>
        </ul>
    </nav>
    <div class="clear"></div>
    <p>Текст...</p>
    <div class="right"></div>
    <div class="clear"></div>
    <p>Текст...</p>
    <hr />
    <h2>Проста сторінка</h2>
    <div id="container">
        <div id="left">
            <h3>Меню</h3>
            <ul>
                <li>
                    <a href="#">Головна</a>
                </li>
                <li>
                    <a href="#">Про нас</a>
                </li>
                <li>
                    <a href="#">Контакти</a>
                </li>
            </ul>
        </div>
        <div id="right">
            <h3>Заголовок статті</h3>
            <p>Текст статті ... Текст статті ... Текст статті ... Текст статті ... Текст
статті ...</p>
```

```
            <p>Текст статті ... Текст статті ... Текст статті ... Текст статті ... Текст статті ...</p>
                        <p>Текст статті ... Текст статті ... Текст статті ... Текст статті ... Текст статті ...</p>
                        <p>Текст статті ... Текст статті ... Текст статті ... Текст статті ... Текст статті ...</p>
            </div>
            <div class="clear"></div>
            <footer>
               <p>Всі права захищені.</p>
            </footer>
         </div>
      </body>
      </html>
```

ФАЙЛ : main.css

```
      .clear {
         clear: both;
      }

      img {
         float: left;
         margin-right: 20px;
      }

      #ul {
         font-size: 200%;
         list-style: none;
         margin: 0;
         padding: 0;
      }
```

```
#ul li {
    float: left;
}

#ul li a {
    border: 1px solid #000;
    padding: 10px 20px;
}

#ul li a:hover {
    background-color: #c00;
}

div.right {
    background-color: #ccc;
    float: right;
    height: 300px;
    width: 300px;
}

h2 {
    text-align: center;
}

#left {
    float: left;
    font-size: 200%;
}

#left h3 {
    margin: 0;
```

```
}

#left {
   margin-right: 20px;
}

#right p {
   margin: 0;
}

footer {
   text-align: center;
}
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

Приклад позиціювання блоків

Приклад. Опрацювати і проаналізувати даний приклад, який складається з двох файлів: index.html і main.css

ФАЙЛ : index.html

```
<html>
<head>
   <title>Вивчаємо CSS</title>
   <link type="text/css" rel="stylesheet" href="css/main.css" />
</head>
<body>
   <div class="top"></div>
   <div class="block_1"></div>
   <div class="block_2"></div>
</body>
</html>
```

ФАЙЛ : main.css

```
body {
    padding-left: 200px;
}

div {
    height: 500px;
    width: 500px;
}

.block_1 {
    background-color: #0c0;
    position: relative;
    left: 50px;
    z-index: 10;
}

.block_2 {
    background-color: #00c;
    position: absolute;
    left: 100px;
    top: 300px;
    z-index: 20;
}

.top {
    background-color: #333;
    height: 100px;
    position: fixed;
    top: 20px;
    width: 100%;
```

```
    z-index: 100;
}
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

ЗАВДАННЯ

Завдання 1.

Виведіть 5 блоків (додавши ширину, висоту і колір фону у кожного) в одну горизонтальну лінію.

Завдання 2.

Виведіть ще 5 блоків нижче, але вже в одну вертикальну лінію (тобто без float).

Завдання 3.

Зробіть горизонтальне меню за допомогою ul і властивості float з 5 посилань. Зробіть їх зовнішній вигляд на свій розсуд, намагаючись використовувати, як можна більше різних властивостей. Обов'язково додайте hover-ефект для посилань.

Завдання 4.

Закріпіть горизонтальне меню зверху екрану (за допомогою position: fixed;). Переконайтеся в цьому, промотавши сторінку вниз (меню завжди має відображатися у верхній частині екрану, незалежно від положення скроллу).