

Лекція 7.

СТВОРЕННЯ МАКЕТА СТОРІНКИ І ВЕРСТКА. МОДУЛІ

Методи компоновання (layout – розмітки) сторінок CSS дозволяють нам контролювати розміщення елементів, які містяться на веб-сторінці. Таке розміщення може завдаватися відносно наступних факторів: положення елемента в звичайному потоці даних за умовчання, положення відносно інших елементів, які розташовані навколо, положення відносно їх батьківського контейнеру (якщо передбачено) та головного вікна перегляду. Існують наступні методи компоновання сторінки:

- Звичайний потік даних (Normal flow)
- Властивості відображення (The display property)
- Еластичний (Flexbox)
- Погоджений (Grid)
- Плаваючий (Floats)
- Позиційний (Positioning)
- Макет таблиці (Table layout)
- Макет з багатьма шпальтами (Multiple-column layout)

Кожна техніка має свою сферу використання, переваги та недоліки. Жодна техніка не призначена для використання окремо. Опанування наведених технік надає можливість вибору найбільш ефективного метода компоновання контенту за виникнення тих або інших потреб або вподобань дизайнера.

7.1. Звичайний потік даних Normal flow

Звичайний потік даних визначає те, як браузер візуалізує HTML-сторінки за замовчуванням, коли не визначено жодного параметра керування компонованням сторінки. Розглянемо наступний приклад HTML:

```
<p>I love my cat.</p>
```

```
<ul>
```

```
<li>Buy cat food</li>
```

```
<li>Exercise</li>
```

```
<li>Cheer up friend</li>
```

```
</ul>
```

```
<p>The end!</p>
```

I love my cat.

- Buy cat food
- Exercise
- Cheer up friend

The end!

Зауваження: Зверніть увагу, що код HTML відображається в тому ж самому порядку, в якому він розташований в початковому коді, з елементами, розташованими один за одним – перший абзац, за яким розташований неупорядкований список, за яким розташований другий абзац.

Елементи, які розташовані один під одним, описуються як елементи *блоку*, на відміну від елементів *рядкових*, які розташовані поруч один з одним на зразок окремих слів у абзаці.

Зауваження: Напрямок, в якому розташовано вміст блочного елемента, описується як Block Direction (напрямок блоку). Властивість Direction CSS задає напрям тексту, комірок таблиць та горизонтальне розташування. Слід використовувати значення *rtl* для мов, які передбачають написання з правого боку ліворуч (іврит або арабська), і значення *ltr* для тих, які передбачають написання з ліва направо (як англійська та більшість інших мов). Напрямок блоку може бути зазначено як вертикальний для мов, які мають горизонтальний режим запису, або як горизонтальний для мов з вертикальним режимом запису, наприклад, японської. Відповідно напрям Inline Direction – це напрям, в якому буде розташовано вміст рядка (наприклад, речення).

Для багатьох елементів на сторінці метод візуалізації звичайного потоку даних є достатнім для компоновання. Як видно з прикладу таке розташування визначається за замовченням (без визначення додаткових параметрів у коді). Для складніших макетів слід змінити розташування за замовченням за допомогою певних інструментів, доступних у CSS.

Нижче неведені методи зміни розташування елементів засобами CSS:

- **Властивості відображення (The [display](#) property)** – Стандартні значення, такі як блок (*block*), строковий елемент (*inline*) або блок в строковому елементі (*inline-block* – значення, яке можна привласнити властивості *display*, яке позначає елемент розташований в рядку, але має важливі властивості блока), можуть змінити поведінку елементів у звичайному потоці даних, наприклад, роблячи елемент блокового рівня подібним до елемента *inline*-рівня. Існують також методи компоновання, які застосовуються через конкретні значення відображення, наприклад, CSS Grid і Flexbox, які змінюють те, як дочірні елементи розміщуються всередині своїх батьківських елементів.

CSS boxes

В загальному випадку в CSS можна виокремити два типи елементів – блочні (*block*) та строкові (*inline*). Такий тип визначає властивості та поведінку елементів в контексті потоку сторінки та відносно інших елементів, розташованих на сторінці.

Якщо елемент визначено як блочний (*block*), то він буде мати наступні властивості:

Починатися завжди з нового рядка.

Буде розширюватися за горизонталлю (вздовж рядку) таким чином, щоб заповнити весь видимий простір контейнера, в якому його розміщено. Як правило це означає, що блок стає такої самої ширини, як контейнер, заповнюючи 100% простору.

До елемента блочного типу можна застосовувати атрибути *width* та *height*.

Елемент блочного типу має зовнішні та внутрішні відступи та рамку, які утворюють простір між ним та іншими елементами навколо нього.

Якщо не змінити відображення елемента блочного типу на строковий тип спеціально, то такі елементи, як заголовки (наприклад, `<h1>`) або абзац `<p>`, використовують тип *block* для відображення візуалізації за умовчанням.

Якщо елемент має тип відображення *inline* (строковий), то:

Такий елемент не буде починатися з нового рядка, а буде продовжувати рядок, в якому його розташовано.

До елемента строкового типу не можна застосовувати атрибути *width* та *height*.

Елемент строкового типу має вертикальні зовнішні та внутрішні відступи та рамку, але вони не утворюють простір між ним на інших елементах навколо нього.

Елемент строкового типу має горизонтальні зовнішні та внутрішні відступи та рамку, які утворюють простір між ним на інших елементах навколо нього.

Елементи `<a>`, який використовується для створення посилань, ``, `` та `` – є прикладами строкових елементів за умовчанням.

Тип відображення, що застосовується до елемента, вказується через значення властивості *display*, такими як *block* та *inline*.

- **Плаваючий (Floats)** — застосування властивості [float](#) визначає тип обтікання даного елемента, наприклад, за значення *left* на блочному рівні інші елементи будуть розташовуватися ліворуч від визначеного такою властивістю елемента.

CSS Demo: float

RESET

float: none;

float: left;

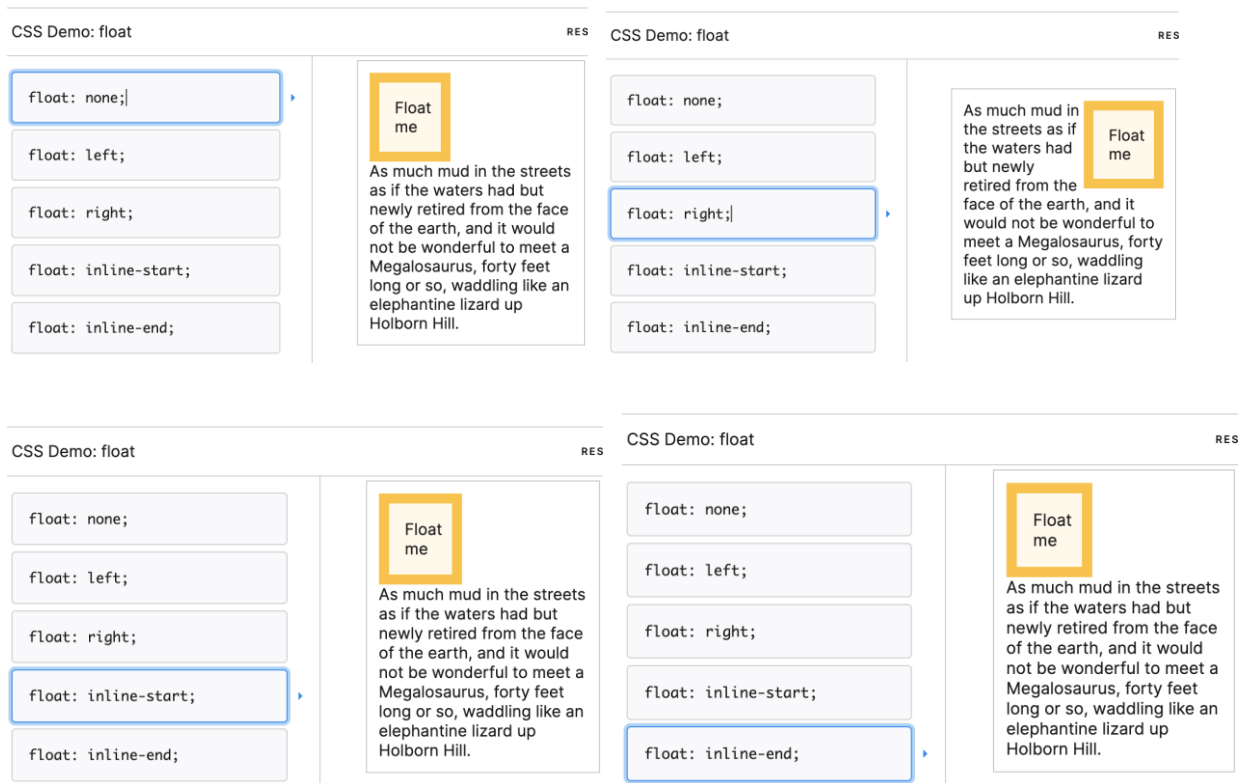
float: right;

float: inline-start;

float: inline-end;

Float me

As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.



- **Властивості позиціонування (The [position](#) property)** — Дозволяє точно визначати розміщення одних елементів (контейнерів) всередині інших. Статичне позиціонування *static* є типовим для нормального потоку даних, але можна змінити розташування таким чином, щоб елементи були розташовані по-різному, за використання інших значень, таких наприклад, як фіксоване розташування *fixed* або вгорі вікна браузера *top of the browser viewport*.
- **Табличні макети (Table layout)** — Функції, призначені для моделювання частин HTML таблиці, можуть бути використані для завдання розташування також і нетабличних елементів за використання параметра *display: table*: та властивостей пов'язаних з цим параметром. Проте таке завдання макету сторінки не рекомендується через ускладнення подальшої модифікації макету.
- **Макет з багатьма шпальтами (Multi-column layout)** — за застосування властивостей такого макета можна утворити візуалізацію, за якої вміст блоку буде розподілений по шпальтах, як у газеті.

7.2. Властивості відображення

Всі основні методи правильного компонування сторінок в CSS передбачають завдання значень для властивості відображення *display*. Ця властивість дозволяє змінити типовий спосіб візуалізації елементів. Всі елементи в звичайного потоку даних мають типові значення властивостей відображення; тобто типовий спосіб візуалізації, якій є налаштованим за умовчанням. Наприклад, абзаци в англійській мові відображаються один під іншим, тому що вони стилізовані з властивістю *display: block*. Якщо ви створюєте посилання навколо якогось тексту всередині абзацу, це посилання залишається в рядку з рештою основного тексту, і не переходить на новий рядок. Це пов'язано з тим, що елемент `<a>` є за умовчанням має властивість *display: inline*.

Можна змінити типову візуалізацію показу. Наприклад, елемент `` типово має властивість *display: block*, тобто елементи списку відображаються один під одним у документі. Якби ми змінили значення відображення на *inline*, то вони б відображались в одному рядку як слова у реченні. Те, що можна змінити значення відображення для будь-якого елемента, означає, що можна підібрати елементи HTML за їх семантичним значенням, не зважаючи на те, як вони будуть виглядати. Тому що те, як вони виглядатимуть завжди можна змінити за допомогою параметрів стилю.

Окрім можливості зміни типової візуалізації, перетворюючи елемент з блокового на строковий і навпаки, існують інші методи компонування елементів, які пов'язані зі значеннями відображення *display*. Однак, під час використання таких методів, як правило, потрібно викликати додаткові властивості. Два значення, що є найбільш важливими для формування макету сторінки, це *display: flex* та *display: grid*.

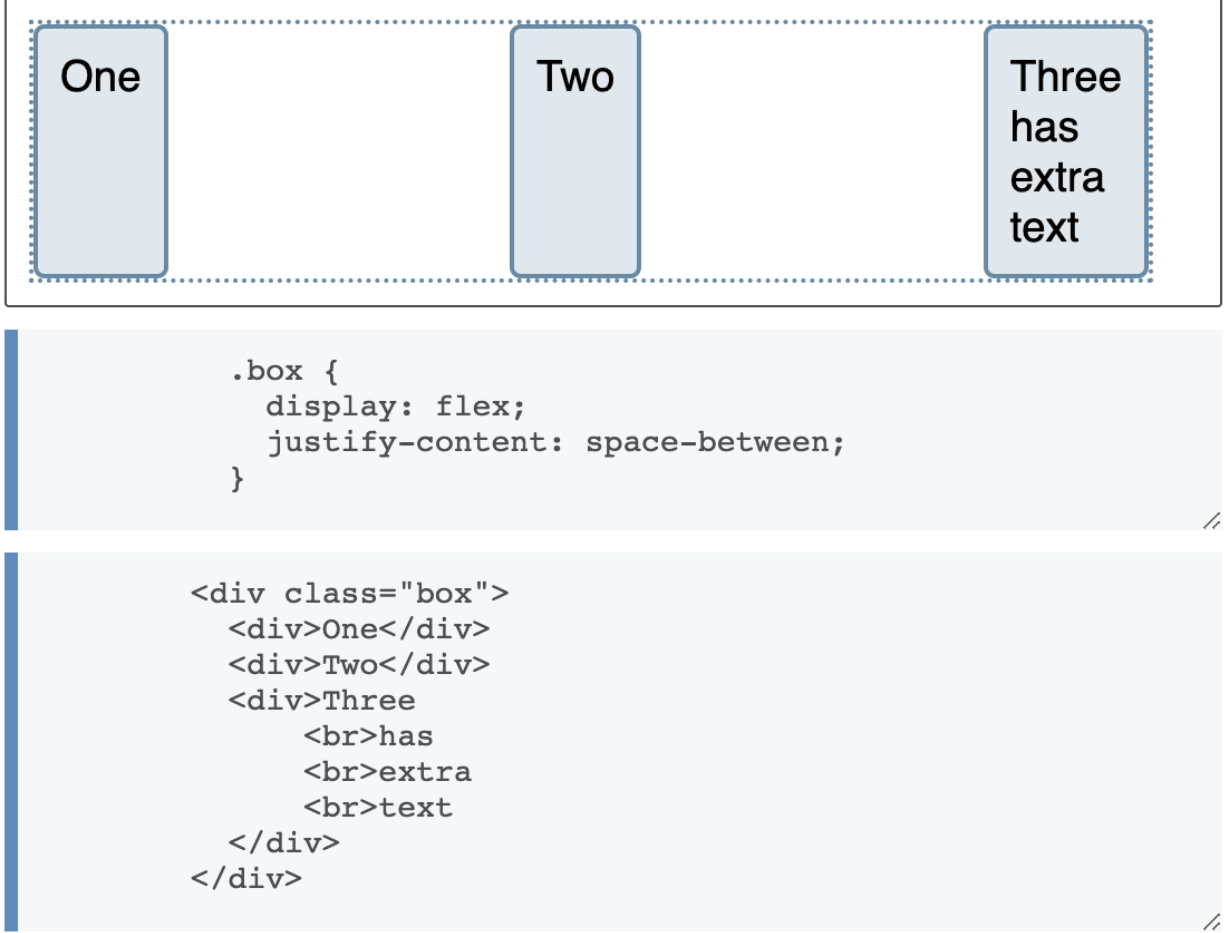
7.3. Макет з еластичними елементами Flexbox

Flexbox – це скорочена назва макета [Flexible Box Layout](#) модуль CSS, розроблений таким чином, щоб спростити розташування елементів в одному фрагменті – у вигляді рядка або стовпчика. Для застосування властивостей *flexbox*, слід зазначити властивість *display: flex* елемента, який є батьківським

для тих елементів, які слід розташувати вказаним чином; в цьому випадку всі елементи, що прямими нащадками так само стають гнучкими елементами.

CSS Flexible Box Layout – це модуль CSS, який визначає модель CSS box, оптимізовану для проектування інтерфейсу користувача, і компоновання елементів в одному вимірі. У моделі Flex, нащадки контейнера Flex можуть бути розміщені в будь-якому напрямку, і можуть змінювати свої розміри, або збільшуватися, щоб заповнити невикористаний простір, або зменшуватися, щоб уникнути переповнення батьківського контейнера. Можна легко маніпулювати вирівнюванням контейнерів-нащадків всередині батьківського вказуючи горизонтальне або вертикальне вирівнювання.

У наступному прикладі для відображення було встановлено *display: flex*, що означає, що три елементи-нащадки набувають властивостей Flex-елементів. Параметр *justify-content* було визначено як *space-between* для того, щоб рівномірно розподілити елементи відносно горизонтальної осі. Елементи-нащадки розташовані на однаковій відстані один від одного, при цьому лівий і правий елементи будуть зливатися з краями батьківського контейнера. Також можна побачити, що елементи мають однаковий розмір стосовно вертикальної осі (однакову висоту), завдяки значенню параметра *align-items*, який за умовчанням визначено як *stretch*. Елементи-нащадки змінюють висоту таким чином щоб дістатися висоти батьківського контейнера.



The diagram illustrates a flex container (dotted line) containing three items (solid boxes). The first box is labeled 'One', the second 'Two', and the third 'Three has extra text'. Below the diagram, the CSS code defines the container's style, and the HTML code shows the structure of the container and its children.

```
.box {  
  display: flex;  
  justify-content: space-between;  
}
```

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three  
    <br>has  
    <br>extra  
    <br>text  
  </div>  
</div>
```

Відображення за значення *display: flex*

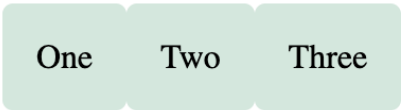
Розмітка HTML, код якої наведено нижче, утворює елемент класу *wrapper*, всередині якого розміщено три елемента `<div>`. За умовчанням ці елементи відображаються як блокові елементи, тобто один під іншим як абзаци.

Проте, якщо застосувати властивість *display: flex* до батьківського елемента, всі три елемента-нащадки будуть розташовані один біля одного. Це пов'язано з тим, що вони наслідують властивості батьківського елемента та набувають властивостей flex-елементів, тобто тих властивостей, які за умовчанням визначають розташування *flexbox* у батьківському контейнері. Вони розміщуються в рядок, оскільки властивість *flex-direction* батьківського елемента має початкове значення *row*. Елементи вирівнюються за висотою, оскільки властивість *align-items* батьківського елемента має початкове

значення *stretch*. Всі елементи вирівнюються за лівим краєм контейнера, залишаючи додаткове місце в кінці рядка.

```
.wrapper {  
  display: flex;  
}
```

```
<div class="wrapper">  
  <div class="box1">One</div>  
  <div class="box2">Two</div>  
  <div class="box3">Three</div>  
</div>
```

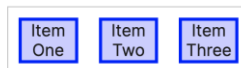


One Two Three

Властивість *flex-direction* задає, як елементи визначені як *flex* елементи розташовуються в контейнері. Ця властивість визначає вирівнювання відносно вертикальної вісі та порядок розташування (прямий або зворотний).

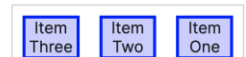
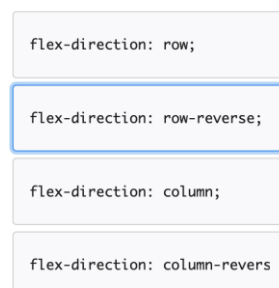
CSS Demo: flex-direction

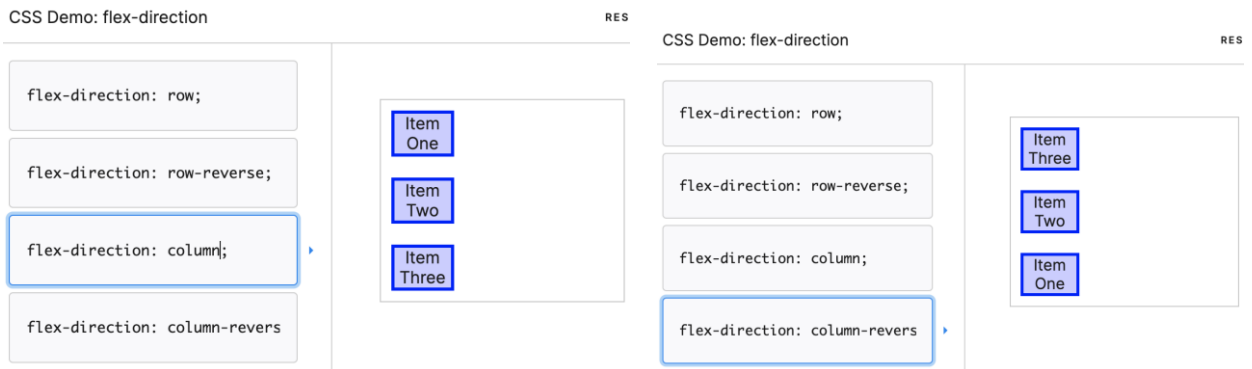
RES



CSS Demo: flex-direction

RES





Налаштування режиму відображення *flex*

Крім властивостей, які можуть бути застосовані до *flex*-контейнера, існують також властивості, які можуть бути застосовані до *flex* -елементів. Ці властивості, серед іншого, можуть змінити спосіб, яким елементи змінюють розміри, дозволяючи їм збільшуватися або зменшуватися відповідно до доступного простору.

Як простий приклад, ми можемо додати властивість *flex* до всіх наших дочірніх елементів, і надати їй значення 1. Це призведе до того, що всі елементи будуть збільшуватися і заповнювати контейнер, а не залишати місце в кінці праворуч. Якщо буде більше місця, то елементи стануть ширшими; якщо буде менше місця, вони стануть вужчими. Крім того, якщо додати до розмітки ще один елемент, інші елементи стануть меншими, щоб вивільнити місце для нового елемента, проте всі елементи разом будуть продовжувати заповнювати весь простір контейнера.

```
.wrapper {  
  display: flex;  
}
```

```
.wrapper > div {  
  flex: 1;  
}
```

```
<div class="wrapper">  
  <div class="box1">One</div>  
  <div class="box2">Two</div>  
  <div class="box3">Three</div>  
</div>
```

One

Two

Three

Наведені інформація є дуже стислим описом властивості *flex*.

7.4. Макет сітки Grid Layout

У той час як властивість *flex* визначає розташування елементів для одновимірного компоновання, макет *Grid Layout* призначений розташування елементів у двох вимірах: за горизонталлю та вертикаллю або у стовпчики та рядки.

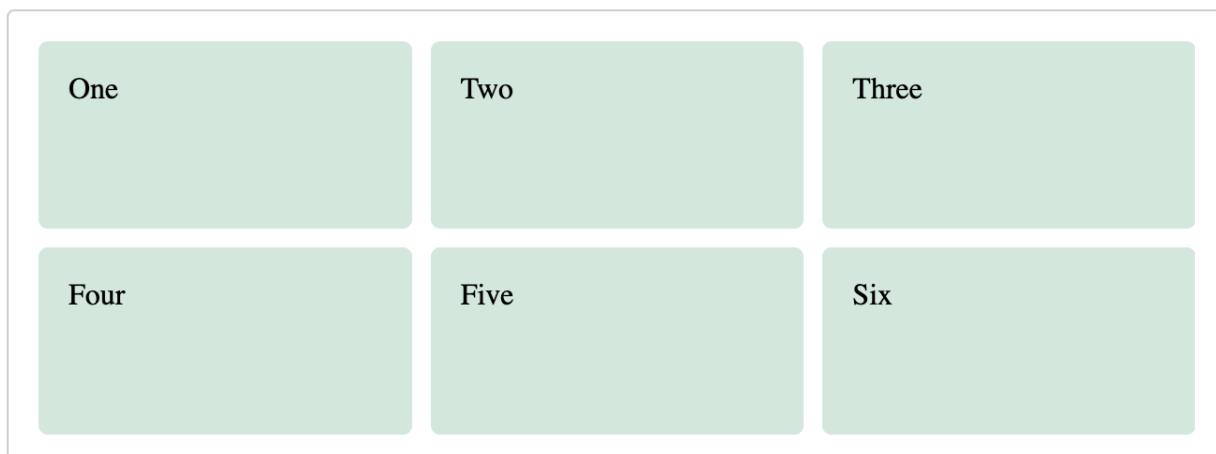
Відображення *display: grid*

Аналогічно до створення елементів *flexbox*, для створення елементів *Grid Layout* слід зазначити відображення як *display: grid*. Наведений нижче приклад використовує розмітку подібну до розмітки для приклада із *flex*-елементами, тобто з одним батьківським контейнером та кількома елементами-нащадками. Крім зазначення параметру *display: grid*, слід також визначити деякі

властивості, що завдають розташування елементів у рядках (за допомогою властивості *grid-template-rows*) та стовпчиках (за допомогою властивості *grid-template-columns*) батьківського контейнера. У прикладі визначено три стовпчики з атрибутом *1fr* (fr – одиниця виміру, яка є часткою (fraction) простору, який є досяжним для елементів в контейнері), а також два рядки 100px. Властивість *gap* завдає відстань між елементами у відсотках або пікселях, якщо вказано одне значення, то відстань буде однаковою за вертикаллю та горизонталлю, якщо два *gap 50px 20px*, то перше значення – відстань за вертикаллю, друге – за горизонталлю. Нам не потрібно завдавати будь-які інші правила розташування елементів-нащадків; вони автоматично розміщуються в клітинках сітки, яку було створено зазначеними атрибутами.

```
.wrapper {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 100px 100px;  
  gap: 10px;  
}
```

```
<div class="wrapper">  
  <div class="box1">One</div>  
  <div class="box2">Two</div>  
  <div class="box3">Three</div>  
  <div class="box4">Four</div>  
  <div class="box5">Five</div>  
  <div class="box6">Six</div>  
</div>
```



Розташування елементів у сітці

Після створення сітки, можна розташувати потрібні елементи безпосередньо в її клітинках, а не покладатися на автоматичне розміщення. У наступному прикладі, визначено таку саме сітку, але цього разу з трьома елементами-нащадками. Ми встановили початковий і кінцевий рядки кожного елемента за допомогою властивостей *grid-column* та *grid-row*. Це призводить до того, що елементи охоплювати декілька доріжок.

```
.wrapper {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 100px 100px;  
  gap: 10px;  
}  
  
.box1 {  
  grid-column: 2 / 4;  
  grid-row: 1;  
}  
  
.box2 {  
  grid-column: 1;  
  grid-row: 1 / 3;  
}  
  
.box3 {  
  grid-row: 2;  
  grid-column: 3;  
}
```

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
</div>
```



Two

One

Three

7.5. Властивість *float*

Створюючи *float* плаваючий елемент ми змінюємо розташування цього елемента та елементів блочного рівня, які сліднують за ним у звичайному потоці даних. Елемент *float* переміщується ліворуч або праворуч і видаляється зі звичайного потоку даних, а навколишні елементи відповідно переміщуються навколо нього.

Властивість *float* має наступні можливі значення:

- *left* — вирівнювання елемента *float* ліворуч, а усі інші елементи обтікають його з правого боку;
- *right* — вирівнювання елемента *float* праворуч, а усі інші елементи обтікають його з лівого боку;
- *none* — визначення елемента *float* не завдано (це значення встановлено за умовчанням);

- *inherit* — визначає, що значення властивості *float* має бути успадковане від елемента-пращура.

У наведеному нижче прикладі елемент `<div>` розташовано ліворуч а всі інші елементи праворуч від нього. Атрибут `margin-right` завдає відступ від елемента праворуч.

Параметр *margin* може завдавати одразу 4 атрибути: *margin-top*, *margin-right*, *margin-bottom* та *margin-left*. Якщо після *margin* вказані лише значення, то вони визначають відступи в наступному порядку: зверху, праворуч, знизу, ліворуч.

```
/* Применяется ко всем четырём сторонам */
margin: 1em;

/* по вертикали | по горизонтали */
margin: 5% auto;

/* сверху | горизонтально | снизу */
margin: 1em auto 2em;

/* сверху | справа | снизу | слева */
margin: 2px 1em 0 auto;

/* Глобальные значения */
margin: inherit;
margin: initial;
margin: unset;
```

Наведемо текст до описаного вище прикладу Це дає нам ефект тексту, який обтікає фіксований ліворуч елемент, і є більшість того, що потрібно знати про елементи типу *float*, які використовуються в сучасному веб-дизайні.

```
<h1>Simple float example</h1>

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


<h1>Simple float example</h1>

<div class="box">Float</div>

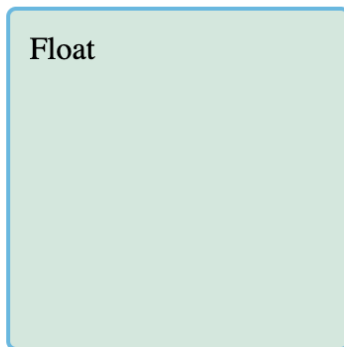
<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

</p>

```
.box {  
  float: left;  
  width: 150px;  
  height: 150px;  
  margin-right: 30px;  
}
```

Simple float example



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit

amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

7.6. Методи позиціонування

Методи позиціонування дозволяють переміщати елемент, з того місця, в якому його було розміщено у звичайному потоці даних в інше місце. Методи позиціонування є методами створення основної розмітки сторінки, за допомогою яких визначають та остаточно налаштовують позиції окремих елементів на сторінці.

Властивість CSS *position* завдає позицію елемента на сторінці: *top*, *right*, *bottom* або *left* та визначає кінцеве розташування на сторінці елементів, які позиціонуються.

CSS Demo: position

RESET

position: static;

position: relative;
top: 40px; left: 40px;

position: absolute;
top: 40px; left: 40px;

position: sticky;
top: 20px;

In this demo you can control the position property for the yellow box.

To see the effect of sticky positioning, select the position: sticky option and scroll this container.

The element will scroll along with its

CSS Demo: position

RESET

position: static;

position: relative;
top: 40px; left: 40px;

position: absolute;
top: 40px; left: 40px;

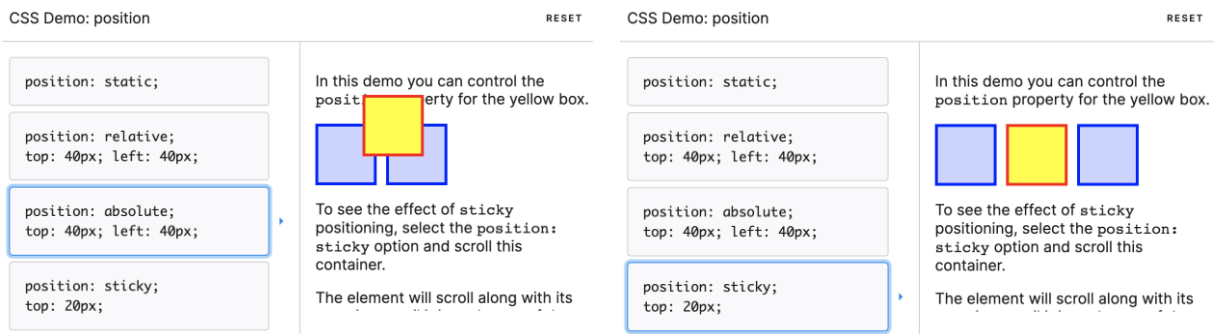
position: sticky;
top: 20px;

In this demo you can control the position property for the yellow box.

To see the effect of sticky positioning, select the position: sticky option and scroll this container.

The element will scroll along with its

18



Існують, однак, корисні техніки для отримання конкретних шаблонів компонування, які покладаються на властивість *position*. Розуміння позиціонування також допомагає краще зрозуміти розташування елементів у звичайному потоці даних та їх переміщення.

Існує п'ять типів позиціонування:

- *Статичне* (Static positioning) є типовим для кожного елемента. Надає елементові розташування за умовчанням в макеті сторінки.
- *Відносне* (Relative positioning) надає змогу змінювати розташування елемента на сторінці, пересуваючи його відносно його положення у звичайному потоці даних, а також надає можливість розташування цього елемента поверх інших елементів.
- *Абсолютне* (Absolute positioning) повністю змінює позицію елемента відносно звичайного потоку компонування сторінки, таким чином ніби він є розташованим на власному окремому шарі. Виходячи з такої позиції можна закріпити даний елемент у положенні відносно країв його найближчого позиційованого батьківського елемента (таким елементом стає `<html>`, якщо не позиціонуються інші). Такий тип позиціонування використовується для створення складних ефектів компонування, таких як панелі вкладок, де різні панелі вмісту розташовані одна за одною та відображаються або приховуються залежно від дій користувача, або інформаційні панелі, які за першого завантаження розташовані на екрані за умовчанням, але передбачають пересування по екрану за допомогою кнопок управління.

- *Фіксоване* (Fixed positioning) є подібним до абсолютного позиціонування, за винятком того, що за допомогою цього метода змінюється розташування елемента відносно вікна браузера, а не іншого елемента. Це корисно для створення ефектів, таких як постійне меню навігації, яке завжди залишається в тому ж місці екрана, під час прокручування іншого вмісту сторінки.
- *Липке* (Sticky positioning) це новий спосіб позиціонування, який визначає елемент як елемент з властивістю *position: relative*, поки він не потрапляє до певної позначки вікна браузера, а після цього його розташування визначається як *position: fixed*.

7.7. Застосування методів позиціонування

Для ознайомлення із застосуванням методів позиціонування розглянемо кілька простих прикладів. Всі приклади мають однакову структуру HTML сторінки: заголовок, за яким слідують три абзаци:

```
<h1>Positioning</h1>

<p>I am a basic block level element.</p>
<p class="positioned">I am a basic block level element.</p>
<p>I am a basic block level element.</p>
```

Така HTML сторінка за умовчанням стилізована наступними CSS параметрами:

```
body {  
    width: 500px;  
    margin: 0 auto;  
}  
  
p {  
    background-color: rgb(207, 232, 220);  
    border: 2px solid rgb(79, 185, 227);  
    padding: 10px;  
    margin: 10px;  
    border-radius: 5px;  
}
```

Застосування цих параметрів визначають наступну візуалізацію:

Positioning

I am a basic block level element.

I am a basic block level element.

I am a basic block level element.

Відносне позиціонування

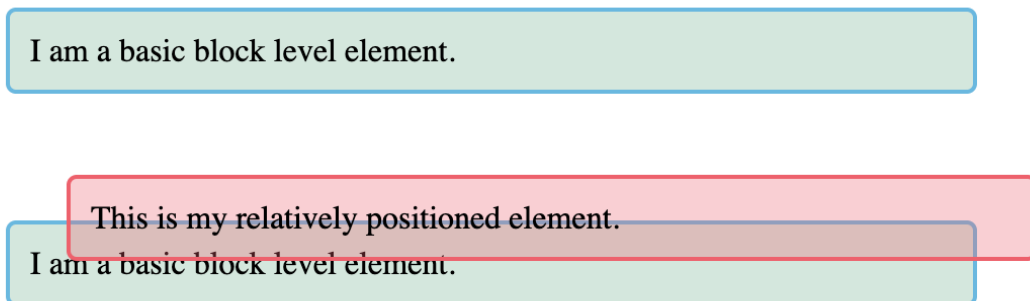
Відносне позиціонування (Relative positioning) дозволяє змінювати розташування елемента відносно його типового положення у звичайному потоці дани. Це означає, що можна реалізовувати такі завдання як переміщення піктограми вниз, щоб вона була в одному рядку з відповідною текстовою міткою. Для цього можна додати наступне правило, щоб додати відносне позиціонування:

```
.positioned {  
    position: relative;  
    top: 30px;  
    left: 30px;  
}
```

В цьому прикладі середньому абзацу було надано параметр позиціонування *position: relative*. Така зміна не призводить до зміни візуалізації, тому слід додати властивості *top: 30px* та *left: 30px*, які вказують у пікселях відстань на яку буде переміщено відповідний елемент вниз та вправо.

Додання описаного коду змінить реалізацію наступним чином:

Relative positioning



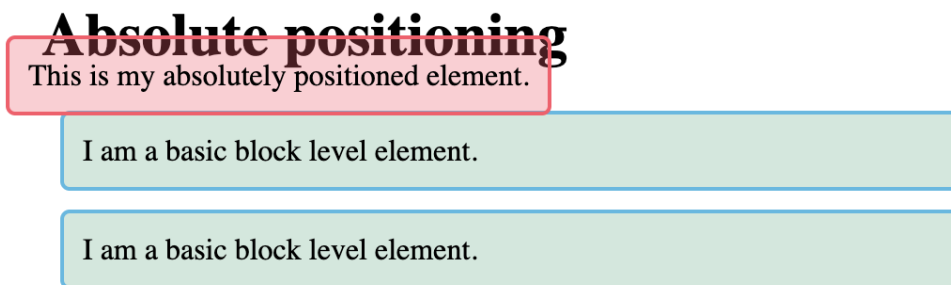
Абсолютне позиціонування

Абсолютне позиціонування (Absolute positioning) використовується для повного видалення елемента зі звичайного потоку даних і замість цього визначає його розташування за допомогою зсувів відносно країв блоку, в якому міститься даний елемент.

Повертаючись до початкового прикладу з відсутністю позиціонування, можна додати наступне правило CSS для реалізації абсолютного позиціонування:

```
.positioned {  
  position: absolute;  
  top: 30px;  
  left: 30px;  
}
```

Тут ми надаємо середньому абзацу позиційний значення `position: absolute` за тих самих властивостей розташування відносно лівого та верхнього країв блоку, що і раніше. Додавання цього коду дасть наступний результат:



Це значно змінює візуалізацію. Елемент, що позиціонується тепер повністю відокремився від решти макету сторінки і розташований поверх нього. Інші два абзаци тепер розташовані разом так, ніби позиціонованого елемента не існує. Властивості *top* та *left* мають інший вплив за абсолютного розташування елементів, ніж той, якого вони мають за відносного розташування. У цьому випадку, зміщення обчислюються зверху і ліворуч від країв контейнера, який цей елемент містить.

Фіксоване позиціонування

Фіксоване позиціонування (Fixed positioning) вилучає елемент зі звичайного потоку даних так само, як і абсолютне позиціонування. Однак,

замість зміщення, що застосовується відносно країв контейнера, застосовується зміщення відносно вікна браузера. Оскільки елемент залишається нерухомим відносно вікна перегляда, можна створювати ефекти, такі як меню, яке залишається фіксованим, коли сторінка прокручується під ним.

Для цього у прикладі HTML документа, що містить три абзаци тексту відповідному класу надаємо властивість *position: fixed*.

```
<h1>Fixed positioning</h1>
```

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

```
<p>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci.

```
</p>
```

```
<p>
```

Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et.

```
</p>
```

```
<p>
```

In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut

id

ornare felis, eget fermentum sapien.

```
</p>
```



```
.positioned {  
  position: fixed;  
  top: 30px;  
  left: 30px;  
}
```

Fixed positioning

Fixed

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci.

Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor

Після прокручування контенту сторінки візуалізація наступна:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci.

Fixed

Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et.

Липке позиціонування

Липке позиціонування (Sticky positioning) є остатнім способом позиціонування, який ми маємо у своєму розпорядженні. Він є сумішшю відносного позиціонування та фіксованого розташування. Коли елемент має властивість *position: sticky*, під час прокручування контенту, що належить до звичайного потоку даних, такий елемент буде прокручуватися разом з іншими елементами потоку поки не сягне певного заданого місця вікна перегляду. Після цього елемент набуває властивостей *position: fixed* і вже не змінює свого розташування (він ніби прилипає до цього місця).

```
.positioned {
  position: sticky;
  top: 30px;
  left: 30px;
}
```

ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Sticky

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem.

Після прокручування контенту сторінки візуалізація наступна:

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor eu lacinia lorem placerat vulputate. Duis felis orci, vulvar id metus

7.8. Макет таблиці

Таблиці HTML є зручним інструментарієм відображення табличних даних, але багато років тому — до того, як навіть базовий CSS почав надійно підтримуватися браузерами — веб-розробники використовували також таблиці для створення макетів веб-сторінок, шляхом розміщення заголовків, нижніх колонтитулів та інших елементів в рядках та стовпчиках таблиць.

Такий метод створення макету є можливим, але має багато недоліків: макети таблиць негнучкі, дуже важкі для розмітки, складні для відлагодження, і семантично неправильні.

Спосіб, у який таблиця виглядає на веб-сторінці за використання розмітки таблиці, обумовлений набором властивостей CSS, які визначають макет таблиці. Ці ж властивості також можуть бути використані для розташування елементів, які не є таблицями, використання такого способу створення макету сторінки іноді називають «використання CSS таблиць».

У наведеному нижче прикладі показано одне таке використання. Слід зазначити, що використання CSS таблиць для компоновання слід вважати застарілим на сьогоднішній день, і його слід використовувати лише для застарілих браузерів, які не підтримують *Flexbox* або *Grid*.

Розглянемо приклад. Спершу розглянемо розмітку, яка створює HTML-форму. Кожен елемент введення даних *input* має *label*. Кожна пара *label/input* розміщена всередині елемента `<div>` для спрощення подальшого компоновання.

```
<form>
  <p>First of all, tell us your name and age.</p>
  <div>
    <label for="fname">First name:</label>
    <input type="text" id="fname" />
  </div>
  <div>
    <label for="lname">Last name:</label>
    <input type="text" id="lname" />
  </div>
  <div>
    <label for="age">Age:</label>
    <input type="text" id="age" />
  </div>
</form>
```

Далі розглянемо код CSS для даного приклада. Більшість з параметрів стилю є досить звичайними, за винятком використання властивості *display* зі значенням *table* для елементів `<form>`, `<div>`, `<label>` та `<input>`. Таким чином ці елементи набувають відображення елементів таблиці: рядків (*table-row*) та комірок (*table-cell*). В основному, вони будуть розміщуватися, як макет таблиці HTML, що призводить до того, що елементи `<label>` та `<input>` будуть вирівнюватися відносно границь комірок за умовчанням. Для остаточного завершення візуалізації слід зазначити відстані між комірками за допомогою атрибуту *margin*.

Абзацу, який містить заголовок надано властивість *display: table-caption* що робить його візуалізацію подібною до елемента `<caption>`, а параметр *caption-side: bottom* завдає розташування цього елемента внизу форми, не зважаючи на те, що в HTML документі цей абзац описаний раніше за опис елементів `<input>`. Це дозволяє отримати ефективну гнучкість візуалізації відносно кода HTML.

```

html {
    font-family: sans-serif;
}

form {
    display: table;
    margin: 0 auto;
}

form div {
    display: table-row;
}

form label, form input {
    display: table-cell;
    margin-bottom: 10px;
}

```

```

form label {
    width: 200px;
    padding-right: 5%;
    text-align: right;
}

form input {
    width: 300px;
}

form p {
    display: table-caption;
    caption-side: bottom;
    width: 300px;
    color: #999;
    font-style: italic;
}

```

Отримуємо наступну візуалізацію:

First name:	<input type="text"/>
Last name:	<input type="text"/>
Age:	<input type="text"/>

Ganna

First of all, tell us your name and age.

7.9. Макет з кількома шпальтами

Макет CSS з кількома шпальтами надає можливість скласти вміст сторінки в кількох колонках, подібно до того, як він виглядає в газетах. Під час читання таких шпальт доводиться прокручувати контент сторінки вгору та

вниз, що деякою мірою ускладнює роботу з веб-документом, проте компонування вмісту у кілька шпальт може бути корисною технікою для вирішення деяких задач.

Щоб надати елементу властивостей макету з кількома шпальтами слід завдати для нього параметр *column-count* із зазначенням кількості шпальт, або параметр *column-width*, який повідомляє браузеру що слід заповнити контейнер якомога більшою кількістю шпальт вказаної ширини.

У наведеному нижче прикладі ми починаємо з блоку HTML всередині елемента з *<div>* класом *container*.

```
<div class="container">
  <h1>Multi-column layout</h1>

  <p>Paragraph 1.</p>
  <p>Paragraph 2.</p>

</div>
```

Використовуємо параметр *column-width 200 px* для цього контейнера, для того, щоб браузер створив кілька 200 піксельних шпальт, стільки скільки вміститься в контейнері а потім розподілити простір, що залишився між створеними шпальтами.

```
.container {
  column-width: 200px;
}
```

Multi-column Layout

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta.

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem.

Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.