

# Web-технології та web-дизайн. CSS

2023/2024  
для студентів 2 курсу

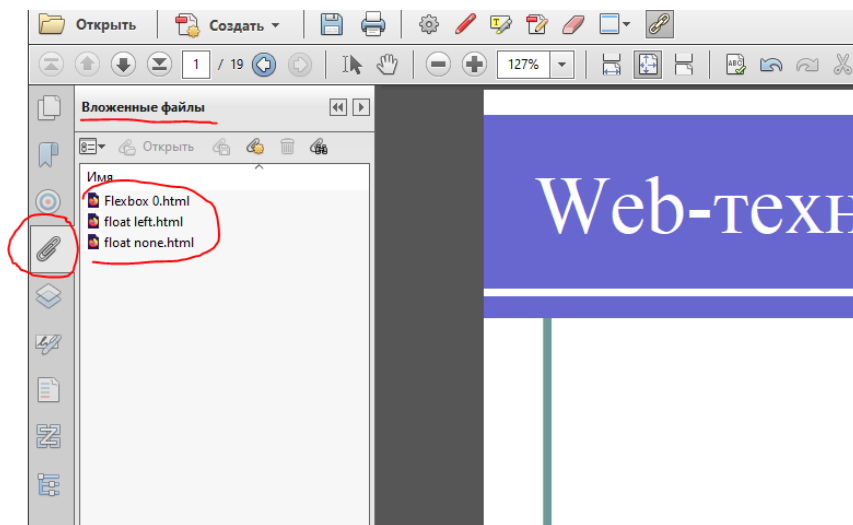
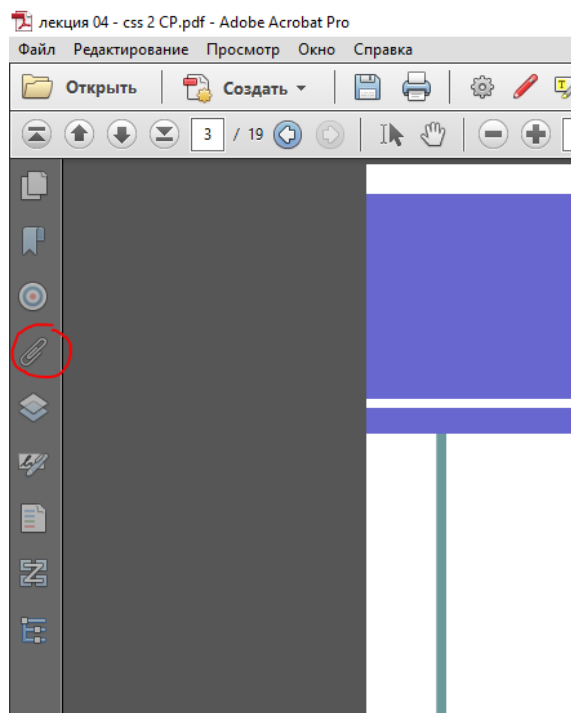
## Самостійна робота до лекції №5

Лекції: старший викладач каф. Штучний інтелект  
Гриньова Олена Євгенівна  
[olena.hrynova@nure.ua](mailto:olena.hrynova@nure.ua)

# Зміст

Підходи до верстки  
Техніки макетування сторінок  
Обтікання  
Семантична верстка  
Скидання стилів браузеру

Перегляньте файли у вкладенні цього файлу.



# ПІДХОДИ ДО ВЕРСТКИ

# Методи верстки

- На основі `<table>` (таблична верстка, застарілий підхід)
- На основі боксів CSS (блочна верстка)
- На основі HTML5 (семантична верстка - підтип блочної верстки)

# Техніки CSS макетування сторінок

- Positioning
- Table layout
- The display property
- Normal flow
- Floats
- Flexbox
- Grid
- Multiple-column layout

Жоден з перерахованих різновидів верстки не втратив своєї актуальності. Більше того, ці методи не використовуються самотійно, а ефективно комбінуються професійними розробниками один з одним.

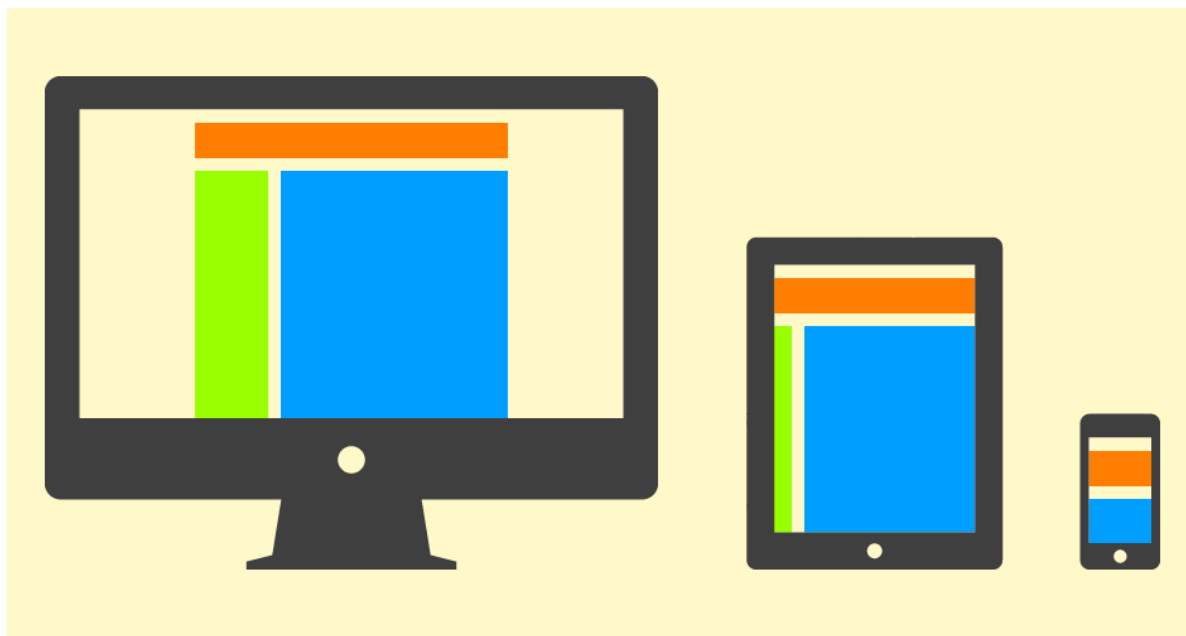
# Типи верстки макетів веб-сторінок

Від виду верстки залежить, як сайт буде відображатися на певному пристрої.

- Статична (static) або фіксована - px
- Гумова (liquid) - %
- Адаптивна (adaptive) - px + media-queries
- Чуйна (responsive) - % + media-queries
- Валідна (відповідність вимоги кросбраузерності)

# Фіксована верстка (static)

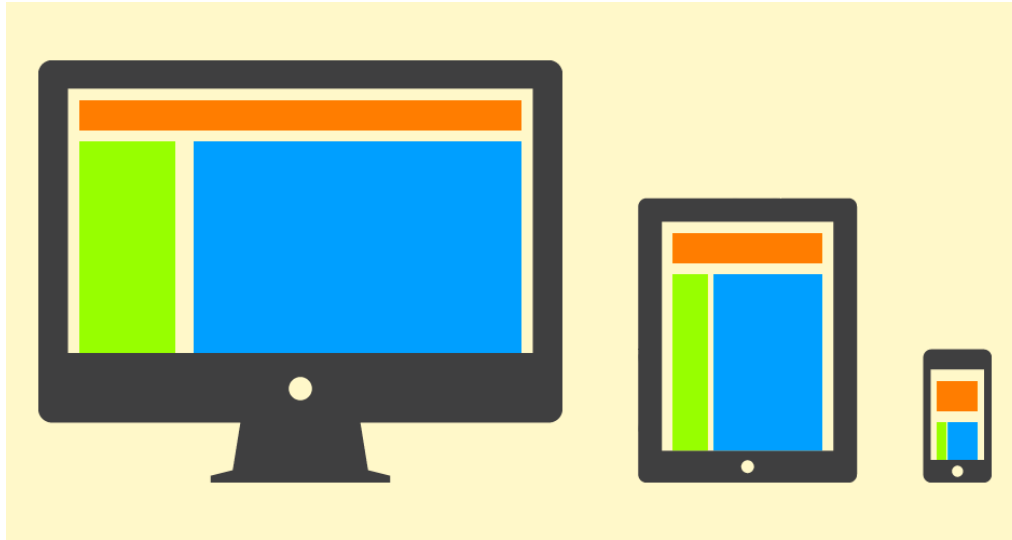
- **Px** (пікселі)
- у всіх елементів є чітко задані параметри, які не змінюються навіть при зміні ширини браузера
- `<style> .static { width:440px } </style>`





# Гумова верстка (liquid)

- %
- макет сторінки, на якому вміст приймає розмір будь-якого екрана за допомогою **відносних показників** для структурних елементів, а не статичних
- здатна підлаштовуватися під різні параметри
- `<style> .liquid { width:50% } </style>`



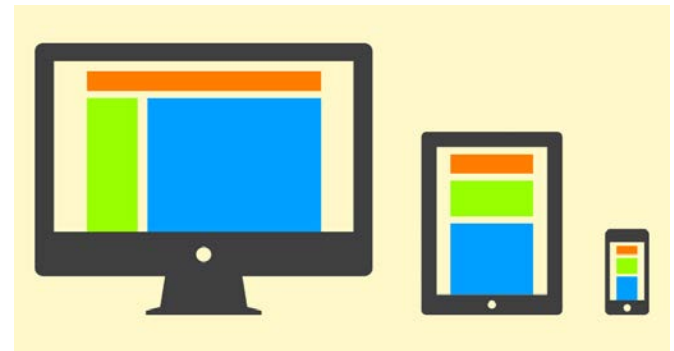
# Адаптивна верстка (Responsive web design)

- px + media-queries
- макет сторінки, що складається з **набору фіксованих макетів** для роботи з **різними роздільними здатностями екранів** и **орієнтації девайсу**
- не розробляти новий дизайн для кожної роздільної здатності, а змінювати розмір і розташування окремих елементів

<style>

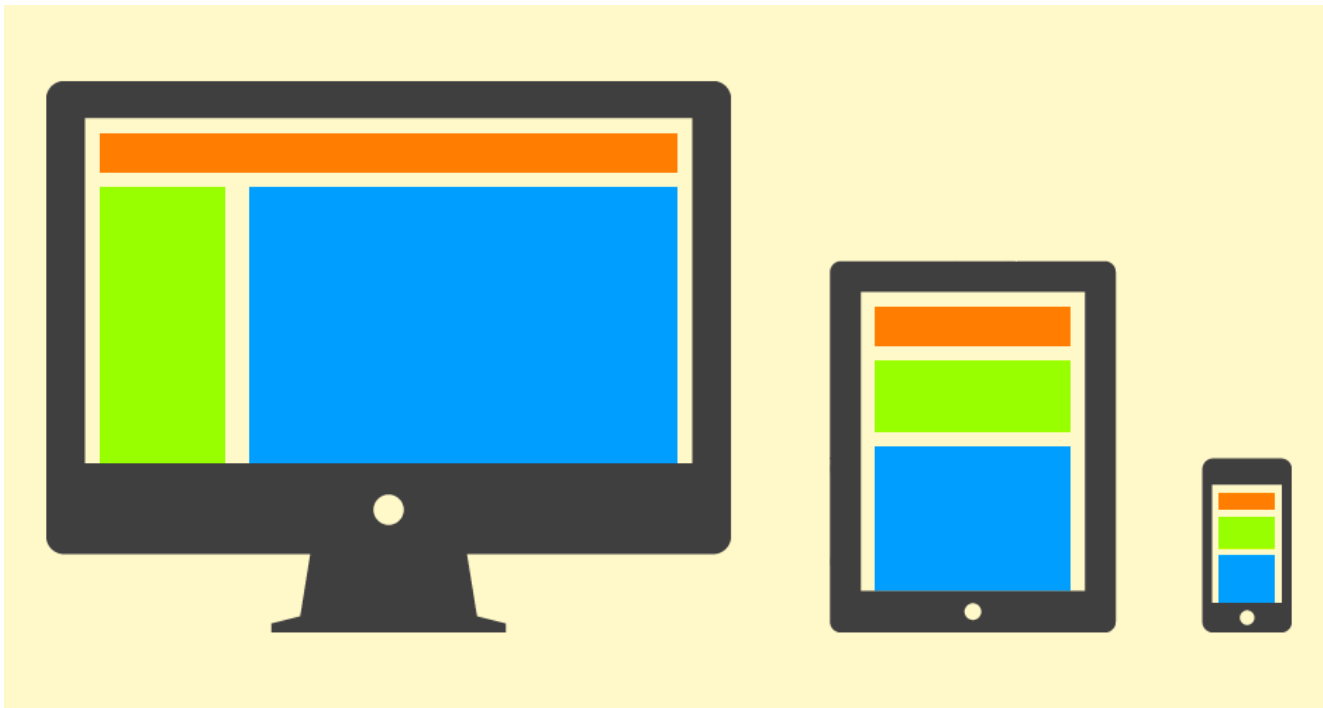
```
@media (max-width: 680px) {  
.adaptive { width: 200px }  
}
```

</style>



# Чуйний макет

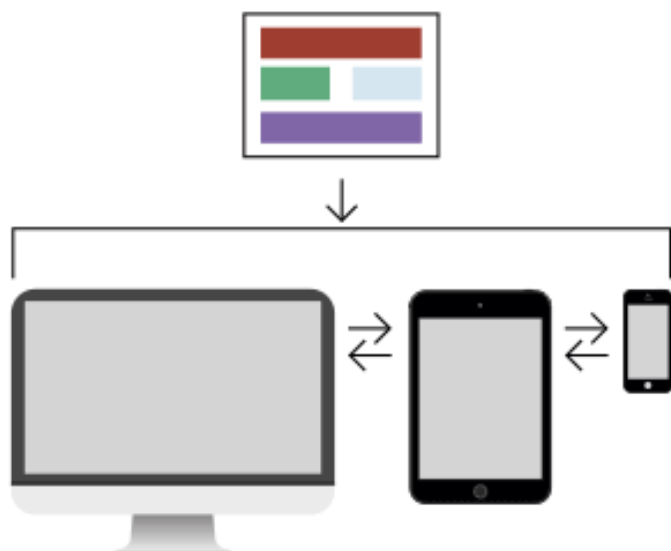
- % + media-queries
- Комбінація адаптивного та гумового макету
- `<style> .responsive { width: 50% } @media (max-width: 960px ){ .responsive { width: 100% } } </style>`



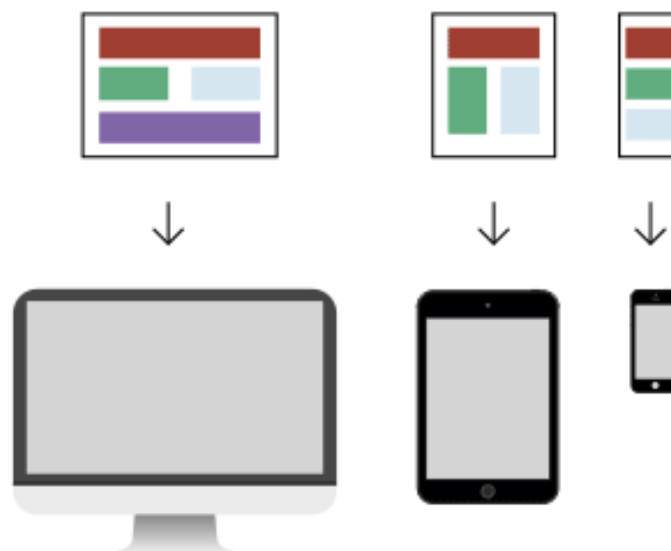
# Чуйний vs адаптивний

**Чуйний дизайн** — один макет для всіх пристроїв,  
**адаптивний дизайн** — один макет для кожного виду пристрою.

Responsive



Adaptive



# Техніка макетування сторінок

# Нормальний потік / базовий потік / normal flow

**Нормальний потік макета** - це система, за допомогою якої елементи розміщуються всередині вікна браузера.

**Елементи на веб-сторінці розміщуються у звичайному потоці, якщо ви не застосували до них жодного CSS, щоб змінити свою поведінку.**

Ви маєте можливість змінити поведінку елементів або змінивши їх положення в цьому звичайному потоці, або видаливши ці елементи з цього потоку.

Добре структурований документ, читабельний в звичайному потоці, є найкращим способом початку будь-якої веб-сторінки.

# Нормальний потік/базовий потік / normal flow

## Як елементи розміщуються за замовчуванням?

Перш за все, окремі бокси елементів розташовуються в залежності від вмісту елементів, потім додаються деякі **padding**, **border** та **margin** навколо них – це боксова модель.

За замовчуванням вміст елемента на рівні бокса становить 100% від ширини його батьківського елемента і такий же високий, як і його контент. Рядкові елементи такі ж високі і широкі, як і їх контент. Ви не можете встановити ширину або висоту на рядкові бокси. Якщо ви бажаєте контролювати розмір рядкового елемента, вам потрібно налаштувати його так, щоб він поведився як блочний елемент за допомогою **display: block;** (або навіть **display: inline-block;**, який змішує характеристики обох).

# Нормальний потік/базовий потік / normal flow

```
body { width: 500px; margin: 0 auto;}  
p { background: rgba(255,255,104,0.3); border: 2px solid rgb(255,84,104); padding: 10px;  
margin: 10px;}  
span { background: white; border: 1px solid black;}
```

<h1>Базовий потік документа</h1>

<p>Я є базовим елементом рівня блоку. Мої сусідні блокові елементи знаходяться на новому рядку піді мною.</p>

<p>За замовчуванням ми охоплюємо 100 процентів ширини нашого батьківського елемента, і ми такі ж високі, як і наш child-контент. Наша загальна ширина та висота – це наш контент + внутрішній відступ (padding) + ширина / висота межі.</p>

<p>Ми відокремлені нашими полями. Через схлопування полів ми відокремлені шириною одного з наших полів, а не обох, та суміжним текстом, якщо є простір. Рядкові елементи, що не влазять <span>переміщуються до нового рядка, якщо це можливо (наприклад, цей текст)</span> якщо це неможливо, вони переходять до нового рядка, подібного до цього зображення:

</p>

Див вкладений файл **1\_Базовий потік документа.html**



# Нормальний потік/базовий потік / normal flow

## Базовий потік документа

Я є базовим елементом рівня блоку. Мої сусідні блокові елементи знаходяться на новому рядку піді мною.

За замовчуванням ми охоплюємо 100 процентів ширини нашого батьківського елемента, і ми такі ж високі, як і наш child-контент. Наша загальна ширина та висота – це наш контент + внутрішній відступ (padding) + ширина / висота межі.

Ми відокремлені нашими полями. Через схлопування полів ми відокремлені шириною одного з наших полів, а не обох, та суміжним текстом, якщо є простір. Рядкові елементи, що не влязять переміщуються до нового рядка, якщо це можливо (наприклад, цей текст) якщо це неможливо, вони переходять до нового рядка, подібного до цього зображення:



# ОБТІКАННЯ

# Обтікання і позиціонування

Обтікання та позиціонування - це CSS-методи для відображення потоку та впорядкування елементів на сторінці. **Позиціонування** - це спосіб завдання елемента, який буде розміщений у будь-якому місці сторінки з точністю до пікселів.

При **обтіканні** елемент зміщається вліво або вправо, а контент розташовується з решти сторін.

# Нормальний потік

В CSS-моделі макета текстові елементи розміщуються зверху вниз у тому порядку, в якому вони з'являються у вихідному коді, та зліва направо.

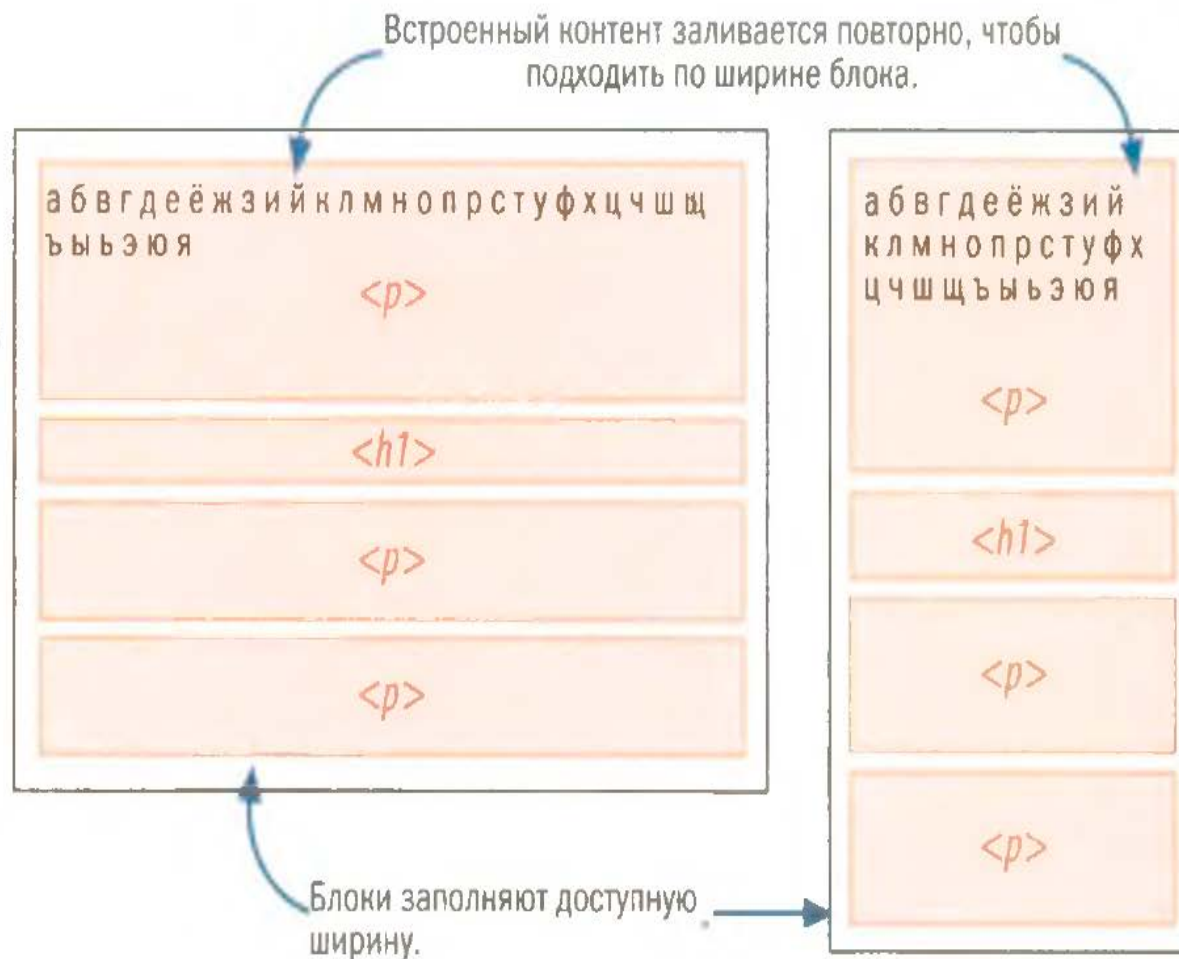
Блокові елементи розташовуються зверху один за одним і заповнюють доступну ширину вікна браузера або іншого елемента, що містить їх. Вбудовані елементи та текстові символи стають в лінію поруч один з одним, щоб заповнити блочні елементи.

Коли перевизначаються розміри вікна або батьківського елемента, блочні елементи розширюються або стискаються до нової ширини, а вбудований контент повторно заповнюється, щоб підходити по розміру.

# Нормальный поток

Блоки выкладываются в том порядке, в котором они появляются в исходном коде.

Каждый блок начинается с новой линии.



# Нормальний потік

**Об'єкти в нормальному потоці** впливають на розташування навколо них, на веб-сторінках елементи не накладаються один на одного і не збиваються в купу, вони створюють «простір» один для одного.

**Обтікання і позиціонування** по-різному змінює взаємозв'язок елементів в нормальному потоці різними способами.

# Обтікання, плаваючі елементи

**Плаваючими** (обтічними) елементами будемо називати такі елементи, які обтікаються по контуру іншими об'єктами веб-сторінки, наприклад, текстом.

Плаваючі елементи досить активно застосовуються при верстці і в основному служать для втілення таких завдань:

- обтікання зображень текстом;
- створення візок;
- розташування шарів по горизонталі (додавання колонок);
- колонки.

# Властивість float

## **float**

Значення: **left** | **right** | **none** | **inherit**

Значення за замовчуванням: **none**

Застосування: до всіх елементів

Успадкування: Ні



# Властивість float

У цьому прикладі застосовується властивість **float** до елемента **img**, щоб перемістити його вправо.

На малюнках показано, як абзац і зображення, відображаються за замовчуванням, і як це виглядає при застосуванні властивості **float**.

```
<p>  Багато хто думає, що Lorem Ipsum - узятий зі стелі псевдо-латинський набір слів, але це не зовсім так. Його коріння сягає одного фрагмента класичної латині 45 року н.е., тобто більше двох тисячоліть тому....</p>
```

Див вкладений файл **2\_Обтікання без CSS.html**

# Обтікання, без CSS

## Обтікання без CSS



Багато хто думає, що Lorem Ipsum - узятий зі стелі псевдо-латинський набір слів, але це не зовсім так. Його коріння сягає одного фрагмента класичної латині 45 року н.е., тобто більше двох тисячоліть тому...

# Обтікання, без float

## Обтікання без float



Багато хто думає, що Lorem Ipsum - узятий зі стелі псевдо-латинський набір слів, але це не зовсім так. Його коріння сягає одного фрагмента класичної латини 45 року н.е., тобто більше двох тисячоліть тому...



Вбудоване зображення в звичайному потоці. Простір поруч із зображенням зберігається вільним

```
p { padding: 15px;  
background-color: #FFF799;  
border: 2px solid #6C4788;}
```

Див вкладений файл  
**[3\\_Обтікання без float.html](#)**

# Обтікання, використовуючи float

## Обтікання float

Багато хто думає, що Lorem Ipsum - узятий зі стелі псевдо-латинський набір слів, але це не зовсім так. Його коріння сягає одного фрагмента класичної латині 45 року н.е., тобто більше двох тисячоліть тому. Річард МакКлінток, професор латині з коледжу Hampden-Sydney, штат Вірджинія, взяв одне з найдивніших слів у Lorem Ipsum, 'consectetur', і зайнявся його пошуками у класичній латинській літературі. В результаті він знайшов незаперечне першоджерело Lorem Ipsum у розділах 1.10.32 і 1.10.33 книги 'de Finibus Bonorum et Malorum' ('Про межі добра і зла'), написаної Цицероном в 45 році н.е. Цей трактат з теорії етики дуже популярний в епоху Відродження. Перший рядок Lorem Ipsum, 'Lorem ipsum dolor sit amet..', походить від одного з рядків у розділі 1.10.32 Класичний текст Lorem Ipsum, який використовується з XVI століття, наведено нижче. Також дано розділи 1.10.32 та 1.10.33 'de Finibus Bonorum et Malorum' Цицерона та їх англійський переклад, зроблений Н. Rackham, 1914 рік.



Вбудоване зображення  
переміщене вправо.  
Зображення переміщається,  
а текст обгортається  
навколо нього

```
img {float: right;}  
p { padding: 15px;  
background-color: #FFF799;  
border: 2px solid #6C4788;}
```

Див вкладений файл  
**4\_Обтікання, з float.html**

# Обтікання зображення, align

Для подібного форматування використовується атрибут **align** тега **<img>** із значенням **left** або **right**. Щоб створити проміжок між символами і краєм зображення до тегу **<img>** також додається атрибут **hspace** для горизонтального відступу і **vspace** для вертикального.

```
<p></p>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
...</p>
```

```
img {float: right;}
```

# Обтікання, floating image

Недолік - відступ зліва і справа задається **однаковим**.

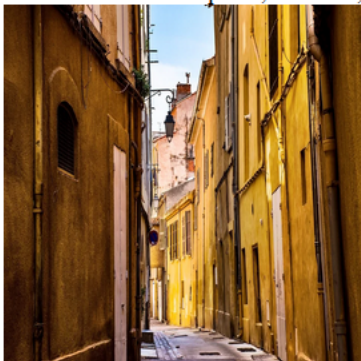
Те ж саме стосується відступу зверху і знизу. Це пов'язано з особливістю атрибутів **hspace** і **vspace**, тому відступ встановлюється через властивості стилю **margin**, а обтікання здійснюється за допомогою **float**.

Див вкладений файл **5\_Обтікання, float align.html**

# Обтікання, floating image

## CSS float властивість

Sed ut perspiciatis, unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam eaque ipsa, quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt, explicabo. Nemo enim ipsam voluptatem, quia voluptas sit, aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos, qui ratione voluptatem sequi nesciunt, neque porro quisquam est, qui dolorem ipsum, quia dolor sit, amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt, ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit, qui in ea voluptate velit esse, quam nihil molestiae consequatur, vel illum, qui dolorem eum fugiat, quo voluptas nulla pariatur?



At vero eos et accusamus et iusto odio dignissimos ducimus, qui blanditiis praesentium voluptatum deleniti atque corrupti, quos dolores et quas molestias excepturi sint, obcaecati cupiditate non provident, similique sunt in culpa, qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio, cumque nihil impedit, quo minus id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et

voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat.

```
<style>  
img {  
  float: left;  
}  
</style>
```

Див вкладений файл **6\_Обтікання, float left.html**

# Поведінка обтічних елементів

Зображення, переміщене зі свого положення в звичайному потоці, все ще продовжує впливати на навколишній вміст. Плаваючі елементи схожі на острови в потоці – вони знаходяться за його межами, але вони повинні обтікати їх. Така поведінка унікальна для обтічних елементів.

Плаваючі елементи залишаються в області вмісту елемента.

Поля зберігаються. Увесь блок елемента між зовнішніми кордонами обтікається.



# Обтікання, створення врізок

**Врізка** - це блок із зображеннями та текстом, вбудованими в основний текст. Врізка зазвичай розташована на лівому або правому краю текстового блоку, а основний текст обтікає її з інших сторін.

Щоб врізка виділялася в тексті, їй зазвичай встановлюють фоновий колір і додають рамку. Своїм зовнішнім виглядом врізка нагадує вищевказаний спосіб обтікання текстом навколо картинки, тому код для створення врізок практично ідентичний попередньому.

```
<style type="text/css">
.incut {
  float: left; /* Обтекание врезки по правому краю */
  width: 100px; /* Ширина врезки */
  background: #fc0; /* Цвет фона */
  padding: 5px; /* Поля вокруг картинки */
  margin: 5px 10px 5px 0; /* Отступы вокруг рисунка */
  border: 1px solid #333; /* Параметры рамки */
}
</style>
```

# Обтікання, створення врізок

```
<body>
```

```
<p class="incut">Ut wisi enim ad minim veniam, quis nostrud exerci taion  
ullamcorper suscipit lobortis nisl.</p>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam  
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat  
volutpat.</p>
```

```
</body>
```

Ut wisi enim  
ad minim  
veniam, quis  
nostrud exerci  
taion  
ullamcorper  
suscipit  
lobortis nisl.

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit, sed diam nonummy nibh  
euismod tincidunt ut laoreet dolore magna  
aliquam erat volutpat.

# Обтікання вбудованих текстових елементів

**Завжди встановлюйте ширину для обтічних текстових елементів.** Без цього область контенту блока розшириться до максимально можливої ширини (або, в деяких браузерах, вона може стиснутися до її мінімально можливої ширини).

**Обтічні вбудовані елементи поведуться як блокові елементи.** Задавши обтікання вбудованому елементу, ви змушуєте його дотримуватися правил відображення блокових елементів, а поля відображаються з усіх чотирьох сторін.

**Поля обтічними елементами не стискаються.** У звичайному потоці, перекриваючі верхні і нижні поля скорочується (перекриваються), але для обтічних елементів поля підтримуються з усіх боків, як зазначено.

# Обтікання блочних елементів

Що відбувається, коли ви переміщуєте блок всередині нормального потоку. У цьому прикладі весь елемент абзацу переміщується вліво.

`<p>Что такое Lorem Ipsum?</p>`

`<p id="float"> "Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit..."</p>`

`<p> Lorem Ipsum - это текст-"рыба", часто используемый в печати и вэб-дизайне. Lorem Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века...</p>`

# Обтікання блочних елементів

```
p#float {  
  float : left ;  
  width : 200px ;  
  margin-top: 0px;  
  background: #A5D3DE;  
}  
p {  
  border: 1px solid red;  
}
```

# Обтікання блочних елементів

Что такое Lorem Ipsum?

"Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit..." "Нет никого, кто любил бы боль саму по себе, кто искал бы её и кто хотел бы иметь её просто потому, что это боль.."

Lorem Ipsum - это текст-"рыба", часто используемый в печати и вэб-дизайне. Lorem Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века. В то время некий безымянный печатник создал большую коллекцию размеров и форм шрифтов, используя Lorem Ipsum для распечатки образцов. Lorem Ipsum не только успешно пережил без заметных изменений пять веков, но и перешагнул в электронный дизайн. Его популяризации в новое время послужили публикация листов Letraset с образцами Lorem Ipsum в 60-х годах и, в более недавнее время, программы электронной вёрстки типа Aldus PageMaker, в шаблонах которых используется Lorem Ipsum.

# Обтікання блочних елементів

Додано червону лінію навколо всіх **p** елементів, щоб показати їх межі, встановлені значення верхнього поля (**margin-top**) у плаваючому елементі дорівнює **0** (нуль), щоб замінити у браузері параметри поля за замовчуванням у абзацах. Це дозволяє обтічному абзацу вирівнятися з верхньою стороною абзацу, розташованого нижче.

Абзац відсувається вліво, а нижче розташований контент обтікає його, навіть не дивлячись на те, що блоки зазвичай складаються зверху один на одного.

**Ви повинні задавати ширину для обтічних елементів блоків.**

**Елементи не переміщуються вище своїх джерел у вихідному коді.**

# Обгортання, горизонтальне розташування шарів

За замовчуванням блокові елементи розташовуються вертикально один під іншим, але за допомогою властивості **float** ви можете зробити їх розташованими поруч по горизонталі. В цьому випадку потрібно встановити ширину шарів і встановити для них **float**.

**Якщо не вказати ширину, вона буде дорівнювати вмісту шару з урахуванням полів і меж.**



# Заборона обтікання

Якщо використовується обтікання елементами, важливо знати, як вимкнути обтікання текстом і повернутися до звичайного макета.

Це робиться шляхом **заборони обтікання елементів**, які розташовані нижче плаваючого. Застосування властивості **clear** до елемента запобігає її появі поруч із обтічним і змушує його розташуватися в найближчому доступному «порожньому» просторі під плаваючим елементом.

# Обтікання, властивість clear

Властивість **clear** - повернення нормального потоку документа.

Очищення **float** відбувається за допомогою властивості **clear**, яке приймає кілька різних значень, найбільш часто використовується значення - **left, right** и **both**.

**clear:** none | left | right | both | inherit

**left** — скасовує обтікання з лівого краю елемента **clear: left**. В цьому випадку всі інші елементи з цього боку будуть опущені вниз, і розташовані під поточним елементом;

**right** — скасовує обтікання з правого боку елемента **clear: right**;

**both** —скасовує обтікання елемента одночасно з правого і лівого краю. Це рекомендується встановлювати, коли потрібно скасувати обтікання елемента, **але невідомо точно з якого боку**.

**none** - скасовує дію властивості **clear**, при цьому обтікання елемента відбувається, як задано за допомогою властивості **float** або інших налаштувань.

# Обтікання, створення двох стовпців

```
<head>
<style type="text/css">
  .layer1 { width: 150px; /* Ширина перш шар */ background: #f0f0f0; /* Цвет фона */
            float: left; /* Обтекание по правому краю */ }
  .layer2 { width: 250px; /* Ширина второго слоя */ background: #fc0; /* Цвет фона */
            float: left; /* Обтекание по правому краю */ }
  .layer1, .layer2 { padding: 7px; /* Поля вокруг текста */ margin-bottom: 1em;
                    /* Отступ снизу */ }
  .layer3 { clear: both; /* Отменяем обтекание */ }
</style>
</head>
<body>
  <div class="layer1"> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam
    nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. </div>
  <div class="layer2"> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam
    nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. </div>
  <div class="layer3"> Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper
    suscipit lobortis nisl ut aliquip ex en commodo consequat. </div>
</body>
```

# Обтікання, створення двох стовпців

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

# Обтікання, створення двох стовпців

Створення стовпців за допомогою **float** має ряд особливостей:

- після плаваючих елементів потрібно додати елемент з властивістю **clear**, яка **вимикає обтікання**. Це необхідно, якщо ви маєте намір використовувати нижче розташовані шари;
- друга особливість пов'язана з представленням плаваючих шарів. Якщо вікно браузера зменшити до певної межі, то шари **переміщуються вертикально**

# Обтікання, властивість `clear`

Щоб скасувати дію обтікання, потрібно додати властивість **`clear`** до елемента, який з'являється після плаваючого. Зазвичай вводять універсальний клас, наприклад, **`clear`**, і вставляють порожній тег **`<div>`** з цим класом.

Іноді виникають комбінації, в яких використання **`clear`** виходить з ладу. Це відбувається, коли в коді одночасно зустрічаються декілька різних плаваючих елементів.

# float проти inline-block

## float



## inline-block



# Техніки CSS макетування сторінок

Двома найбільш важливими для задач CSS макетування сторінок є **display: flex** та **display: grid**.



# Flexbox

Flexbox (скорочено від Flexible Box Layout) - це модуль, призначений для полегшення макетування в **одному** з вимірів - як рядок або як стовпець. Для використання, встановіть властивість **display:flex** для батьківського елемента тих елементів, до яких хочете застосувати цей тип макетування, всі його прямі нащадки стануть **flex-елементами**.

# Grid Layout

В той час, коли Flexbox призначений для одновимірної розмітки, **Grid Layout** призначений для двовимірної — розміщуючи об'єкти у рядки та стовпці.

# Table layout

Те, як таблиця виглядає на веб-сторінці при використанні розмітки таблиці, обумовлено набором властивостей CSS, які визначають макет таблиці. Ці властивості можуть використовуватися для розміщення елементів, які не є таблицями, таке використання іноді описується як «використання **CSS таблиць**».

Використання таблиць CSS для макетування слід вважати застарілим методом на даний момент, за винятком ситуацій, коли у вас **старі браузер**и без підтримки **Flexbox** або **Grid**.

# Table layout

```
form {  
  display: table; margin:0 auto;}  
form div {  
  display: table-row; }  
form label, form input {  
  display: table-cell;  
  margin-bottom: 10px; }  
.....
```

Имя:	<input type="text"/>
Фамилия:	<input type="text"/>
Возраст:	<input type="text"/>

*Прежде всего, сообщите нам свое имя и возраст.*

```
<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>
```

# Multi-column layout

**Multi-column layout** дає нам спосіб розташовувати контент в стовпцях, подібно до того, як текст розташовується в газеті. Хоч і читання стовпців вгору і вниз менш корисно в контексті вебу, так як ви не хочете змушувати користувачів прокручувати вгору і вниз, розміщення контенту по стовпцях може бути корисною технікою.

Щоб перетворити блок на контейнер із декількома стовпцями використовують властивість **column-count**, яка говорить браузеру скільки колонок ми хочемо мати, або властивість **column-width (en-US)**, яка говорить браузеру заповнити контейнер якомога **більшою кількістю стовпців**, принаймні, такої ширини.

# Семантичне макетування

# Семантичне макетування

Семантика в макетуванні - це відповідність тегів до інформації, розташованой всередині них. Семантика коду також досягається шляхом зменшення кількості тегів. Таким чином, ми створюємо чистий, читабельний, валідний HTML код. Така сторінка буде **швидше завантажуватися і ранжуватися пошуковими системами.**

HTML-розмітка і назви CSS-класів повинні відображати не оформлення, а **сенси.**

# Семантичне макетування

## Неправильно

```
<div style=" color : red ;  
border : 1 px solid red">  
Плохая вёрстка сообщения об  
ошибке: атрибут style!  
</div>
```

## Правильно

```
<div class=" error">  
Сообщение об ошибке (error),  
правильная вёрстка!  
</div>
```



# Семантичне макетування

- Заголовки повинні бути позначені **h1, h2, h3, h4**, але ніяк не **b** чи **strong**.
- При створенні меню краще за все це використовувати **ul** список, всередині якого будуть лежати **li**-елементи меню. Цим ми показуємо, що посилання **рівносильні**. Якщо є пункти другої вкладеності, відповідно створюємо всередині первинного **li**-елемента ще один **ul**-список.
- Всі службові зображення (іконки, стрілки, ...) повинні бути записані в коді CSS. У HTML тег **img** слід використовувати лише для великих зображень. (розмір яких більший ніж 100 x 100 px).


# Семантичне макетування

- Параграф блоку тексту створюється за допомогою **p** тега, але ніяк не **div**.
- Не використовувати атрибут **style** всередині HTML тега. **Всі стилі виносимо в окремий CSS файл.**
- Те ж саме стосується і JavaScript.
- Дотримуватись ієрархії та логіки документа. **Більш важливі елементи сторінки повинні бути на початку HTML-коду, менш важливі - в кінці.** За допомогою стилів CSS і **div** блоків цього не складно досягти з будь-якою шаблонною схемою.


# Семантичне макетування

```
<div>Heading here</div>
<div>Posted in: <a href="#">Web Design</a></div>
<div>Content here...</div>
```

Погано




Heading here  
Posted in: [Web Design](#)  
Content here...




```
<h2>Heading here</h2>
<p>Posted in: <a href="#">Web Design</a></p>
<p>Content here...</p>
```

Добре



**Heading here**  
Posted in: [Web Design](#)  
Content here...



# Скидання стилів браузера

# Скидання стилів браузера

```
/* reset browser styles */  
html, body, div, span, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a,  
abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small,  
strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form, label,  
legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas, details,  
embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby, section,  
summary, time, mark, audio, video      {margin: 0;  
      padding: 0;  
      border: 0;  
      font-size: 100%;  
      vertical-align: baseline;}  
article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section  
{display: block;}  
body    {line-height: 1.2;}  
ol       {padding-left: 1.4em; list-style: decimal;}  
ul       {padding-left: 1.4em; list-style: square;}  
table    {border-collapse: collapse; border-spacing: 0;}  
/* end reset browser styles */
```

# Скидання стилів браузера

Немає такого поняття, як елемент HTML без стилів. Кожна веб-сторінка використовує принаймні один CSS - **стиль клієнтської програми**.

Цей файл CSS включений в браузер і викликається:

- кожного разу, коли відображається веб-сторінка;
- до того, як застосовується будь-який з наших CSS.

**Де зберігаються стандартні таблиці стилів у Google Chrome?**

<https://chromium.googlesource.com/chromium/blink/+/master/Source/core/css/html.css>

В епоху сучасної веб-розробки скидання стилю не так необхідне або зовсім не потрібно, оскільки проблеми сумісності CSS в браузерах зараз набагато рідше, ніж вони були в часи старого доброго IE6.

**Зробіть розумне скидання стилів браузера за замовчуванням!**

# Література

## Книги



Дякую за увагу!