**Прості правила розмітки**

Розмітка сторінки може бути непростим заняттям, особливо спочатку. Який тег вибрати? Який клас додати? Якою має бути розмітка, щоб стилі одного елемента не поламали інший?

На всі ці випадки є прості правила, дотримуючись яких ви зможете легко писати чистий, добре структурований HTML-код, який буде зручно читати та приємно підтримувати.

1. Використовуйте HTML-теги за змістом

**Елементи для основної розкладки**

* header – шапка сторінки чи блоку.
* footer – підвал сторінки чи блоку.
* main – головна смислова частина сторінки.
* section – розділи всередині основного контенту.
* article – окрема стаття, пост або коментар.
* nav – навігація, посилання для переміщення сайтом.
* aside – бічна колонка, додатковий контент, що не входить до main.

**Елементи для вмісту**

* h1-h6 – заголовки. Зазвичай h1 – це назва сайту. Заголовки потрібно використовувати як ієрархію, це важливо для доступності.
* ul і ol – списки, в них зручно розміщувати будь-які елементи, що перераховуються.
* button – кнопка, наприклад, елемент керування або кнопка для надсилання форми.

Для елементів без особливого значеннєвого навантаження можна використовувати div або span.

Для розмітки сторінки не можна використовувати теги, призначені для оформлення тексту: наприклад, b та i. Вони мають власні стилі, які згодом може знадобитися перевизначити чи скинути — простіше відразу вибрати елемент, який не має стилів за замовчуванням. Тег p доречно використовувати для блоків тексту, але для інших випадків краще вибрати div.

Це далеко не всі теги, які існують: тут є зручний список тегів, згрупованих за змістом, з коментарями та прикладами коду (<http://html5doctor.com/#glossary>).

**Доступність**

Доступність – це дуже важлива річ. Будь-яка людина може мати проблеми зі здоров'ям або просто опинитися в незручних умовах, коли не вдається користуватися сайтом звичним способом. Тому доступність – це про зручність користування інтерфейсом для всіх.

Наприклад, правша зламав праву руку і намагається рухати мишу лівою – йому буде дуже незручно та незвично користуватися інтерфейсом, з яким раніше не було жодних проблем. Або ви в автобусі, що бовтається, по дорозі на роботу намагаєтеся читати з телефону і не промахуватися по кнопках. Або жінка з малюком на руках намагається одночасно утримати малюка та набрати щось одним пальцем — будуть проблеми одночасно і з увагою, і з координацією рухів.

Безумовно, є люди, які через стан свого здоров'я постійно відчувають такі труднощі, але також є й інші, для яких цей стан є тимчасовим, і таких людей набагато більше. Іноді це ми самі. Тому доступність – це важливо.

Доступність – це просто

Забезпечити доступність на елементарному рівні можна за допомогою семантичної розмітки, тобто використовуючи теги за змістом. Це важливо з двох причин:

* Якщо не завантажаться стилі, користувач отримає не кашу з тексту та картинок, а зрозумілу сторінку, яку легко зможе прочитати.
* Скрінрідери зможуть розрізнити та прочитати всі елементи на сторінці. Зустрівши посилання, вони скажуть, що це посилання і на нього можна натиснути, список зачитають як список, а не як набір розрізнених тегів, вони прочитають описи для картинок і побудують структуру сторінки, використовуючи заголовки.

І все це просто за рахунок використання тегів за призначенням.

Ось тут можна почитати як скрінрідери бачать сайти і що вони можуть там не знайти, якщо сторінка зверстана погано (<http://css.yoksel.ru/inaccessibility/>).

Найпростіші приклади:

1. Посилання та кнопки

Якщо потрібно зробити клікабельний елемент, виберіть посилання або кнопку. Вибирайте посилання, якщо клік виводить на іншу сторінку, якщо ні – використовуйте кнопку. Скріндрідери розуміють ці елементи як активні, і можуть озвучити це для користувача. Якщо замість кнопки використовується div або span, скрінрідер не зрозуміє, що можна натиснути на нього.

Також, по можливості, робіть клікабельну область великого розміру навіть якщо елемент візуально невеликий. Це особливо важливо для мобільних версій, де ми клацаємо пальцем, але і на широких екранах буде зручніше, якщо за кнопкою або посиланням можна потрапити не прицілюючись.

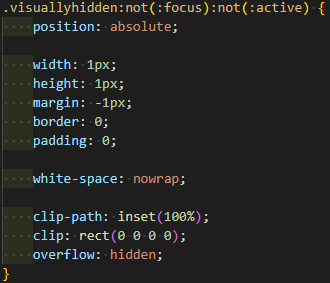
1. Заголовки

Теги заголовків h1-h6 потрібні не тільки для краси, але і для вибудовування структури сторінки, за допомогою яких можна сформувати ієрархічне дерево документа з розділами та підрозділами.

Вибір рівня заголовка на основі ієрархії документа вирішує одразу два завдання:

* Як верстальнику, вам не доведеться ламати голову над тегом для заголовка: у сусідніх елементів рівень заголовків однаковий, якщо у батька заголовок h2, то у дочірніх елементів мають бути заголовки h3 і так далі.
* Ґрунтуючись на заголовках скрінрідери будують структуру сторінки, за якою можна навігуватися, таким чином користувачі читалок можуть відразу вибрати потрібний розділ без необхідності читати весь текст – це як швидко знайти потрібний розділ у змісті книги.

Іноді буває, що на сторінці є самостійний розділ, але за макетом у нього немає заголовка. Виходить, що ця частина сторінки не буде представлена ​​у змісті, яким користуються читалки. Проблему можна вирішити, додавши заголовок, який потім буде прихований за допомогою CSS. Приховувати рекомендується таким кодом (джерело):



Такого заголовка не буде на сторінці, але скрінрідери прочитають його легко. Також це вирішує завдання з тим, що за специфікацією у section і article мають бути заголовки: якщо вони не передбачені за макетом, просто додайте приховані.

1. Зображення

У всіх img на сторінці повинен бути вказаний атрибут alt: його можуть прочитати скрінрідери, його побачать користувачі в яких картинка або відключена або не завантажилась.

Якщо в якості контентного зображення використовується інлайновий SVG, його вміст можна зробити доступним для скрінрідерів, додавши role="img" aria-label="Опис картинки".

Якщо зображення використовуються як тло для тексту, встановіть фоновий колір, схожий на кольори фонового зображення. Якщо картинка не завантажується (або відключена), текст опиниться на констрастному тлі і його все одно можна буде прочитати.

1. Елементи форми

Для розмітки форми обов'язково використовуйте відповідні елементи, наприклад fieldset, legend, label, input та textarea.

Скрінрідери розуміють такі елементи. Вони бачать fieldset як групу інпутів, а legend – як назву групи. Скрінрідер прочитає лейбли як назви для чекбоксів та полів, і користувач зможе вибрати бажані опції або ввести текст.

Без лейблів скрінрідер не зможе зрозуміти призначення інпутів, а без fieldset і legend – зрозуміти як групуються елементи форми і як вони пов'язані між собою, і форма може бути повністю недоступною.

Як бачите, тут немає нічого складного, і просто використовуючи теги за змістом, можна отримати не тільки акуратний осмислений код, але й зробити сторінку трохи кращою з точки зору доступності.

1. Використовуйте у назвах класів прості та короткі слова

Слова, які часто використовуються в CSS-класах:

* Зображення
* image, img, picture, pic — зображення
* icon — значок
* logo – логотип
* userpic, avatar — юзерпік, маленька картинка користувача
* thumbnail, thumb — мініатюра, зменшене зображення
* Текст
* title, subject, heading, headline, caption – заголовок
* subtitle – підзаголовок
* slogan – слоган
* lead, tagline — лід-абзац у тексті
* text – текстовий контент
* desc – опис, варіант текстового контенту
* excerpt — уривок тексту, який зазвичай використовується перед посиланням «Читати далі...»
* quote, blockquote – цитата
* snippet – приклад коду
* link – посилання
* copyright, copy – копірайт
* Списки
* list, items — список
* item – елемент списку
* Блоки
* page – кореневий елемент сторінки
* header – шапка (сторінки або елемента)
* footer – підвал (сторінки або елемента)
* section — розділ контенту (один із кількох)
* main, body – основна частина (сторінки або елемента)
* content – вміст елемента
* sidebar – бічна колонка (сторінки або елемента)
* aside – блок з додатковою інформацією
* widget – віджет, наприклад, у бічній колонці
* Розкладка
* wrapper, wrap – обгортка, зазвичай зовнішня
* inner – внутрішня обгортка
* container, holder, box – контейнер
* grid — розкладка (сторінки або елемента) у вигляді сітки (зазвичай містить у собі row та col)
* row – контейнер у вигляді рядка
* col, column – контейнер у вигляді стовпця
* Елементи управління
* button, btn – кнопка, наприклад, для відправлення форми
* control — елемент керування, наприклад, стрілки «Вперед/назад» у фотогалереї, кнопки керування слайдером
* dropdown — список, що випадає
* Медіавирази
* phone, mobile — мобільні пристрої
* phablet – телефони з великим екраном (6-7")
* tablet – планшети
* notebook, laptop — ноутбуки
* desktop – настільні комп'ютери
* Розміри
* tiny – маленький, крихітний
* small – невеликий
* medium – середній
* big, large – великий
* huge – величезний
* narrow – вузький
* wide – широкий
* Різне
* search – пошук
* social – блок іконок соцмереж
* advertisement, adv, commercial, promo — рекламний блок (не рекомендується використовувати такі класи для блоків із внутрішньою рекламою)
* features, benefits – список основних особливостей товару, послуги
* slider, carousel — слайдер, інтерактивний елемент із прокручуванням вмісту
* pagination – посторінкова навігація
* user, author — користувач, автор запису чи коментаря
* meta – блок з додатковою інформацією, наприклад, блок тегів та дати в пості
* cart, basket — кошик
* breadcrumbs – навігаційний ланцюжок, «хлібні крихти»
* more, all — посилання на повну інформацію
* modal – модальне (діалогове) вікно
* popup – спливаюче вікно
* tooltip, tip – спливаючі підказки
* preview — анонс, уривок, наприклад новини чи посту, може складатися із заголовка, опису та картинки. Передбачається посилання на повну версію
* overlay — рівень, що перекриває, наприклад, для затемнення зображень або створення модальних вікон
* Стани
* active, current — активний елемент, наприклад поточний пункт меню
* visible – видимий елемент
* hidden – прихований елемент
* error – статус помилки
* warning – статус попередження
* success – статус успішного виконання завдання
* pending — стан очікування, наприклад, перед зміною статусу error або success

1. Використовуйте БЕМ для іменування класів

БЕМ розшифровується як "Блок Елемент Модифікатор". Насправді це цілий стек технологій, з якого ми скористаємося лише угодою по іменуванню класів (<https://ru.bem.info/methodology/naming-convention/>).

Чому БЕМ?

* БЕМ дозволяє створювати абсолютно незалежні блоки. Блоки та елементи отримують унікальні імена, тому стилі для одного елемента нічого не поламають в іншому.
* БЕМ допомагає легко вигадувати будь-яку кількість класів, що не повторюються між собою.
* БЕМ допомагає писати код, що самодокументується, в класі будь-якого елемента міститься інформація про нього.

БЕМ (Блок, Елемент, Модифікатор) – компонентний підхід до веб-розробки. У його основі лежить принцип поділу інтерфейсу на незалежні блоки. Він дозволяє легко та швидко розробляти інтерфейси будь-якої складності та повторно використовувати існуючий код, уникаючи «Copy-Paste».

БЕМ-сутностями називаються блоки, елементи та модифікатори.

Це поняття може застосовуватися як приватне, якщо розглядається окрема БЕМ-сутність і як збірне для блоків, елементів і модифікаторів.

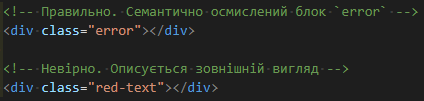
1. Блок

Логічно та функціонально незалежний компонент сторінки, який можна повторно використати. HTML блоки представлені атрибутом class. Блок інкапсулює поведінку (JavaScript), шаблони, стилі (CSS) та інші технології реалізації.

*Особливості:*

* Назва блоку характеризує сенс («що це?» — «меню»: menu, «кнопка»: button), а не стан («який, як виглядає?» — «червоний»: red, «великий»: big).

Приклад.



* Блок не повинен впливати на своє оточення, тобто блоку не слід задавати зовнішню геометрію (у вигляді відступів, кордонів, що впливають на розміри) та позиціонування.
* У CSS БЕМ також не рекомендується використовувати селектори за тегами або id.

Таким чином, забезпечується незалежність, при якій можливе повторне використання або перенесення блоків з місця на місце.

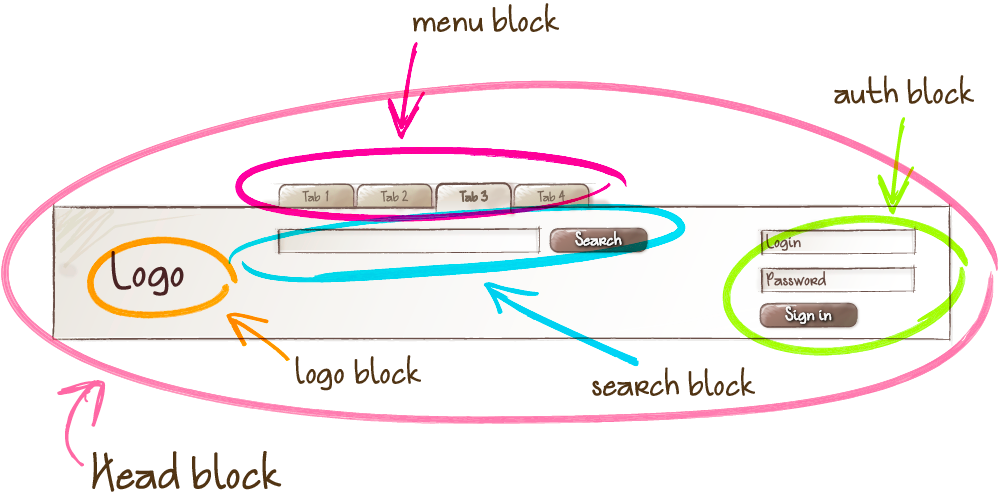
*Принципи роботи з блоками*

* Вкладена структура
* Вільне переміщення
* Повторне використання

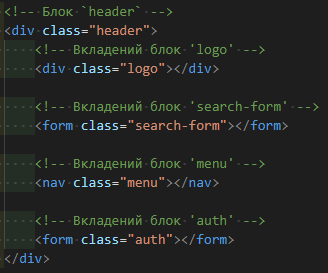
Вкладеність

* Блоки можна вкладати один в одного.
* Допустима будь-яка вкладеність блоків.

Наприклад, блок head може містити логотип (logo), форму пошуку (search) та блок авторизації (auth).



Приклад.



Вільне переміщення

Блоки можна переміщувати в межах однієї сторінки та різних проектів. Незалежність блоку дозволяє змінювати його положення на сторінці та забезпечує коректну роботу та зовнішній вигляд.

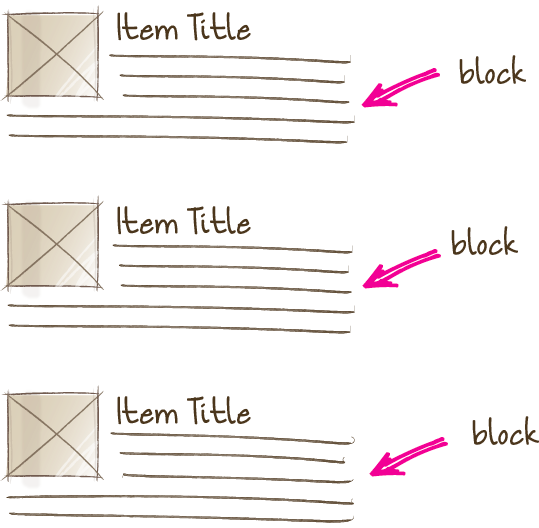
Так, наприклад, логотип та форму авторизації можна поміняти місцями. При цьому вносити зміни до CSS або JavaScript код блоків не потрібно.





Повторне використання

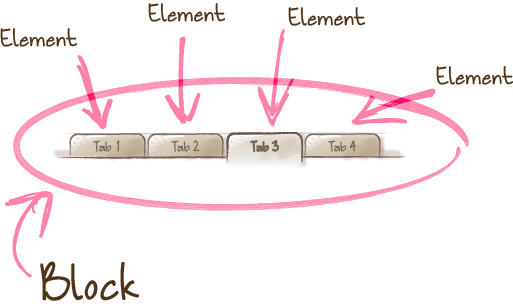
В інтерфейсі може одночасно бути декілька екземплярів одного і того ж блоку.



1. Елемент

Складова частина блоку, яка не може використовуватися у відриві від нього. Елементи не існують поза блоком. Кожен елемент може належати лише одному блоку.

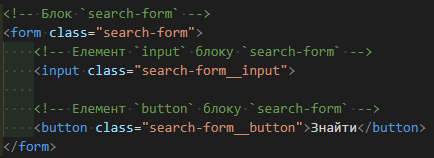
Наприклад, пункт меню поза блоком меню не використовується, отже є його елементом.



*Особливості:*

* Назва елемента характеризує сенс («що це?» — «пункт»: item, «текст»: text), а не стан («який, як виглядає?» — «червоний»: red, «великий»: big).
* Структура повного імені елемента відповідає схемі: ім’я-блоку\_\_ім’я-елемента. Ім'я елемента відокремлюється від імені блоку двома підкресленнями (\_\_).

Приклад.



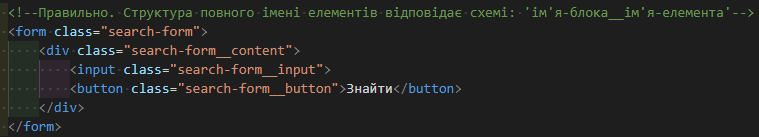
*Принципи роботи з елементами*

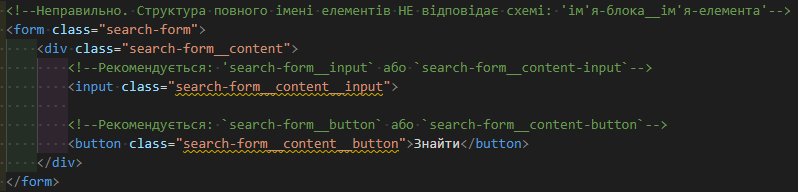
* Вкладеність
* Приналежність
* Необов'язковість

Вкладеність

* Елементи можна вкладати один в одного.
* Допустима будь-яка вкладеність елементів.
* Елемент завжди частина блоку, а не іншого елемента. Це означає, що в назві елементів не можна прописувати ієрархію типу block\_\_elem1\_\_elem2.

Приклад.

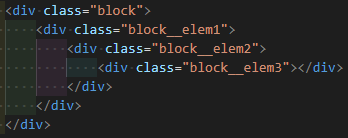




Ім'я блоку задає простір імен, що гарантує залежність елементів від блоку (block\_\_elem).

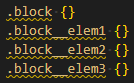
Блок може мати вкладену структуру елементів у DOM-дереві:

Приклад.



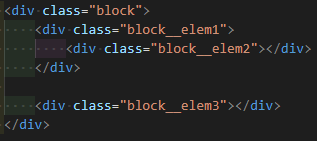
Однак ця ж структура блоку в методології БЕМ завжди буде представлена пласким списком елементів:

Приклад.



Це дозволяє змінювати DOM-структуру блоку без внесення змін до коду кожного окремого елемента:

Приклад.

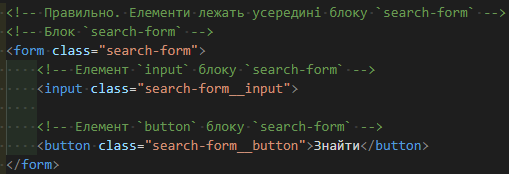


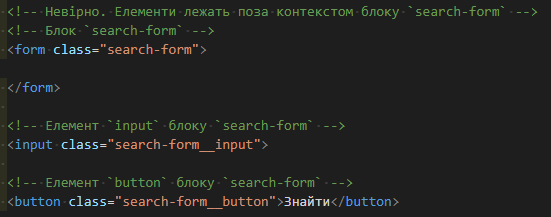
Структура блоку змінюється, а правила елементів та їх назви залишаються колишніми.

Приналежність

Елемент завжди частина блоку і не повинен використовуватися окремо від нього.

Приклад

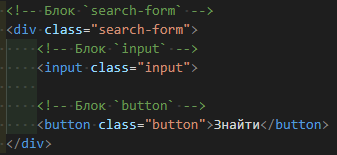




Необов'язковість

Елемент – необов'язковий компонент блоку. Не у всіх блоків мають бути елементи.

Приклад.



*Коли створювати блок, коли елемент?*

* Створюйте блок

Якщо фрагмент коду може бути використаний повторно і не залежить від реалізації інших компонентів сторінки.

* Створюйте елемент

Якщо фрагмент коду не може використовуватись самостійно, без батьківської сутності (блоку).

Виняток становлять елементи, реалізація яких для спрощення розробки вимагає поділу на більш дрібні частини — піделементи. У БЕМ-методології не можна створювати елементи елементів. В такому випадку замість елемента необхідно створювати службовий блок.

1. Модифікатор

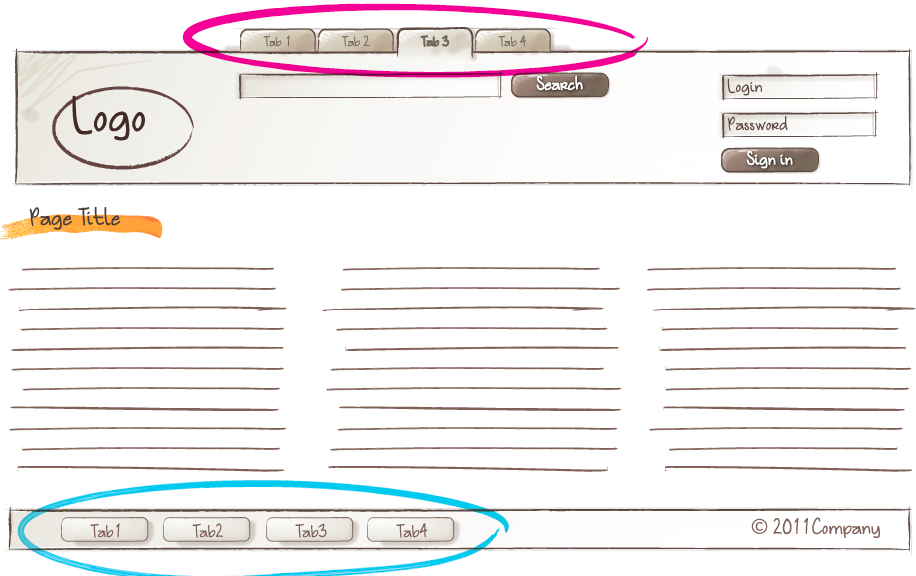
БЕМ-сутність, що визначає зовнішній вигляд, стан та поведінку блоку або елемента. Використання модифікаторів є опціональним, кількість — необмежена. Блоку або елементу не можна одночасно призначити різні значення модифікатора.

За своєю суттю модифікатори схожі на атрибути HTML. Один і той самий блок виглядає по-різному завдяки застосуванню модифікатора.

*Особливості:*

* Назва модифікатора характеризує зовнішній вигляд («який розмір?», «яка тема?» тощо — «розмір»: size\_s, «тема»: theme\_islands), стан («що відрізняється від інших?» — «відключено»: disabled, «сфокусований»: focused) і поведінку («як поводиться?», «як взаємодіє з користувачем?» – «напрямок»: directions\_left-top).
* Ім'я модифікатора відокремлюється від імені блоку або елемента одним підкресленням (\_).

Наприклад, зовнішній вигляд блоку меню (menu) може змінюватись залежно від застосованого модифікатора.



Модифікатори можуть змінюватися в процесі роботи блоку (наприклад, як реакція на DOM-події блоку) та за запитом інших блоків.

Наприклад, при натисканні на кнопку Sign In (DOM-подія click), у разі неправильно заповнених полів Login або Password, на прихований блок повідомлень про помилки встановлюється модифікатор (visible).

*Типи модифікаторів*

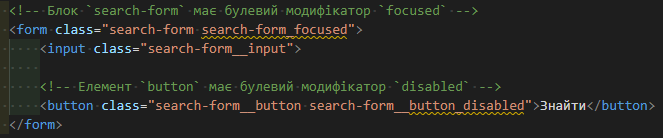
* Булевий

Використовують, коли важливо лише наявність чи відсутність модифікатора, яке значення несуттєво. Наприклад, "відключено": disabled. Вважається, що за наявності булевого модифікатора у сутності його значення дорівнює true.

Структура повного імені модифікатора відповідає схемі:

* ім'я-блока\_ім'я-модифікатора;
* ім'я-блока\_\_ім'я-елемента\_ім'я-модифікатора.

Приклад.



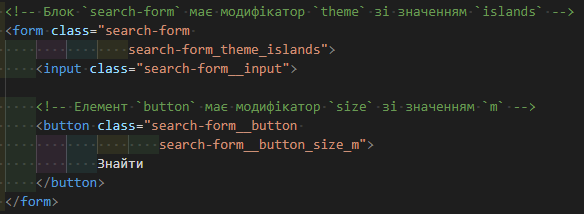
* Ключ-значення

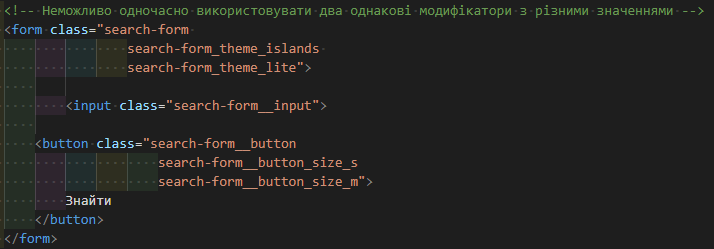
Використовують, коли важливе значення модифікатора. Наприклад, "меню з темою оформлення islands": menu\_theme\_islands.

Структура повного імені модифікатора відповідає схемі:

* ім'я-блоку\_ім'я-модифікатора\_значення-модифікатора;
* ім'я-блока\_\_ім'я-елемента\_ім'я-модифікатора\_значення-модифікатора.

Приклад.

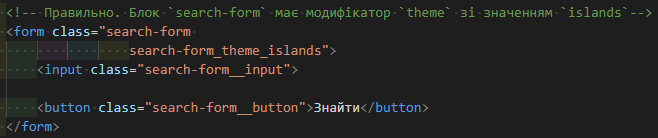


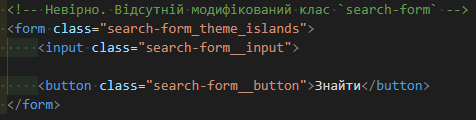


Принципи роботи з модифікаторами – Модифікатор не можна використовувати самостійно

З точки зору БЕМ-методології модифікатор не може використовуватися у відриві від блоку або елемента, що модифікується. Модифікатор повинен змінювати вигляд, поведінку чи стан сутності, а не замінювати її.

Приклад.





1. Мікс

Прийом, що дозволяє використовувати різні БЕМ-сутності на одному DOM-вузлі. Мікси дозволяють:

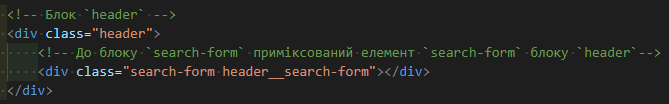
* поєднувати поведінку та стилі декількох сутностей без дублювання коду;
* створювати семантично нові компоненти інтерфейсу з урахуванням наявних.

Розглянемо приклад міксу блоку та елемента іншого блоку.

Припустимо, у проекті посилання реалізовано блоком link. Необхідно зробити посиланнями пункти меню. Існує кілька способів:

* Створити модифікатор для пункту меню, який перетворить пункт на посилання. Але в такому разі для реалізації модифікатора доведеться скопіювати поведінку та стилі блоку link. Це призведе до дублювання коду.
* Скористатися міксом універсального блоку link та елемента link блоку menu. Мікс двох БЕМ-сутностей дозволить застосувати базову функціональність посилань із блоку link та додаткові CSS-правила із блоку menu без копіювання коду.

Приклад.



У цьому прикладі ми поєднали поведінку та стилі блоку search-form та елемента search-form блоку header. Такий підхід дозволяє нам задати зовнішню геометрію та позиціонування в елементі header\_\_search-form, а сам блок search-form залишити універсальним. Таким чином, блок можна використовувати в будь-якому іншому оточенні, тому що він не специфікує жодних відступів. Це дозволяє нам говорити про його незалежність.

1. Файлова структура

Прийнятий у методології БЕМ компонентний підхід застосовується до організації проектів у файловій структурі. Реалізації блоків, елементів та модифікаторів поділяються на незалежні файли-технології, що дозволяє нам підключати їх опціонально.

*Особливості:*

* Один блок – одна директорія.
* Імена блоку та його директорії збігаються. Наприклад, блок header – директорія header/, блок menu – директорія menu/.
* Реалізація блоку поділяється на окремі файли-технології. Наприклад header.css, header.js.
* Директорія блоку є кореневою для піддиректорій відповідних елементів і модифікаторів.
* Імена директорій елементів починаються з подвійного підкреслення (\_\_). Наприклад, header/\_\_logo/, menu/\_\_item/.
* Імена директорій модифікаторів починаються з одинарного підкреслення (\_). Наприклад, header/\_fixed/, menu/\_theme\_islands/.
* Реалізації елементів та модифікаторів поділяються на окремі файли-технології. Наприклад, header\_\_input.js, header\_theme\_islands.css.

На самоопрацювання:

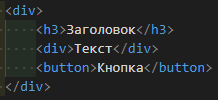
* Угода за назвою (<https://ru.bem.info/methodology/naming-convention/>)
* HTML по БЕМ-у (<https://ru.bem.info/methodology/html/>)
* CSS по БЕМ-у (<https://ru.bem.info/methodology/css/>)
* JavaScript по БЕМ-у (<https://ru.bem.info/methodology/js/>)
* Організація файлової структури ([https://ru.bem.info/methodology/filestructure/](https://ru.bem.info/methodology/filestructure/#%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%BE%D0%B9-%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D1%8B))
* Рівні перевизначення (<https://ru.bem.info/methodology/redefinition-levels/>)
* Способи зміни блоку (<https://ru.bem.info/methodology/block-modification/>)
* Складання БЕМ-проекту ([https://ru.bem.info/methodology/build/](https://ru.bem.info/methodology/build/#%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0-%D0%91%D0%AD%D0%9C-%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0))
* Декларації у БЕМ (<https://ru.bem.info/methodology/declarations/>)
* Вирішення поширених проблем веб-розробки за допомогою БЕМ (<https://ru.bem.info/methodology/solved-problems/>)

**Нижче наведено приклади коду.**

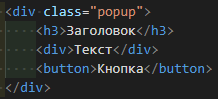
1. Простий приклад: Блок + Елемент

Припустимо, у вас є блок із заголовком, текстом і кнопкою всередині, наприклад, це спливаюче вікно – попап.

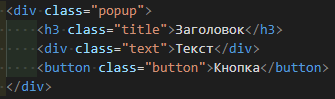
Розмітка:



Додаємо клас, що містить призначення елемента: .popup:

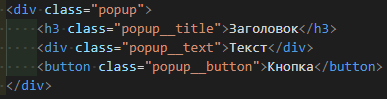


Тепер спробуємо додати класи вкладеним елементам:



Класи зручні, але не унікальні. Якщо на сторінці будуть ще елементи з класами .title та .text, їх стилі можуть торкнутися елементів у попапі. Селектор типу .popup .title може у майбутньому створити проблеми зі специфічністю. Можна вигадати інші класи, але чим більше схожих за змістом елементів, тим складніше вигадувати нові класи.

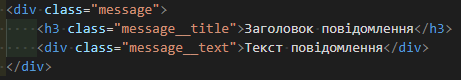
А тепер застосуємо БЕМ-нотацію: кожному елементу всередині блоку додамо префікс із класом батька, наприклад, для заголовка це буде popup\_\_title:



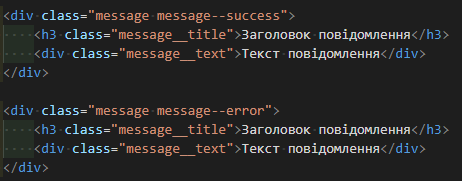
Тепер ці класи легко вирішують відразу два завдання: по-перше, завдяки унікальним класам стилі для них ніколи не перетнуться з іншими подібними елементами на сторінці, а по-друге, за такими класами відразу видно, що це елементи блоку .popup.

1. Приклад складніше: Блок + Елемент + Модифікатор

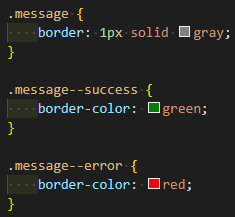
Наприклад візьмемо сервісне повідомлення на сайті. Зазвичай такі повідомлення бувають різних видів, наприклад, повідомлення про успішне завершення або помилку.



Логічно використовувати ту саму розмітку, але з різними колірними темами. Саме тут знадобляться модифікатори.



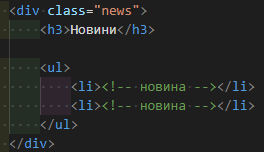
Обом елементам можна додати однакові стилі, використовуючи загальний клас .message і так само легко можна додати окремі стилі для кожного з них, використовуючи унікальний клас з модифікатором:



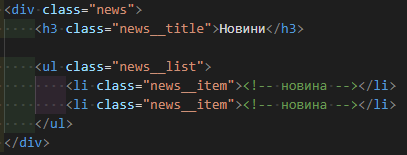
Обидва повідомлення матимуть рамку завтовшки один піксель, але для повідомлення про успішну операцію вона буде зеленою, а для повідомлення про помилку — червоною.

1. Ще важче: що робити, якщо хочеться зробити елемент елемента?

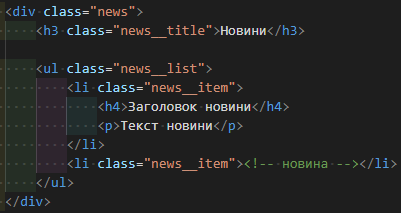
Наприклад, на сторінці є блок новин:



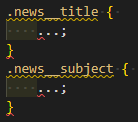
Заголовок блоку логічно отримує клас .news\_\_title, список – .news\_\_list, а окрема новина – .news\_\_item:



Тут жодних проблем виникнути не має. Тепер додамо розмітку окремої новини:



Нам потрібно додати клас заголовку новини. Насамперед спадає на думку .news\_\_title, але такий клас вже зайнятий. Припустимо, що другий елемент буде не .title, а .subject, тоді CSS виходить таке:

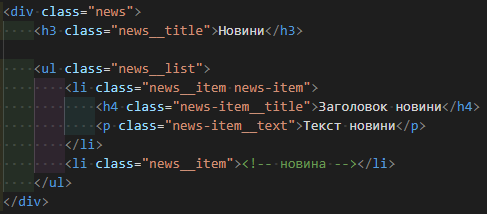


Без додаткових коментарів буде неможливо зрозуміти який із них є заголовком всього блоку, а який — окремої новини. Тому такий варіант не підходить.

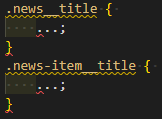
Наступний варіант – .news\_\_item\_\_title, але в БЕМ не можна створювати елемент елемента, і це зрозуміло, тому що виходить каша.

Ще варіант: .news\_\_item-title – теж не годиться, тому що може бути неочевидним як title співвідноситься з item.

Як же бути? Рішення просте: лише на рівні елемента .news\_\_item можна оголосити новий блок (наприклад, .news-item), і будувати вкладені класи від нього. Так, це не самостійний блок, що перевикористовується, тут оголошення блоку потрібно тільки для того, щоб розвантажити селектори. Що виходить:



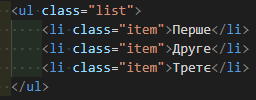
Проблема вирішена: нам більше не потрібно використовувати монструозні класи, при цьому клас точно описує елемент, і в CSS буде відразу зрозуміло, який клас за що відповідає:



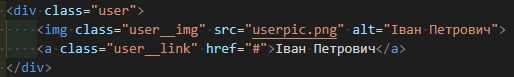
Простий та зручний вихід із незручної ситуації.

**Приклади використання**

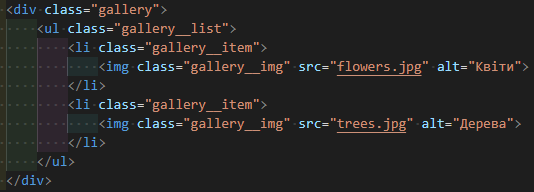
Простий список



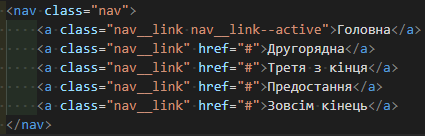
Картинка користувача (юзерпік)



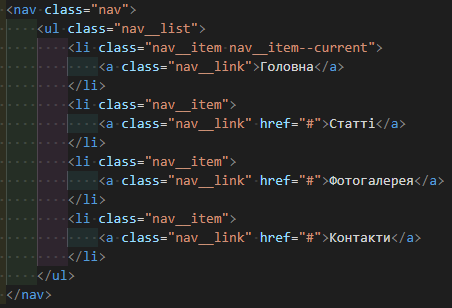
Галерея



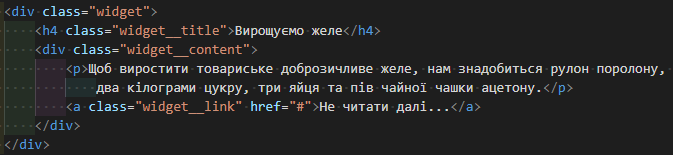
Навігація (простий варіант)



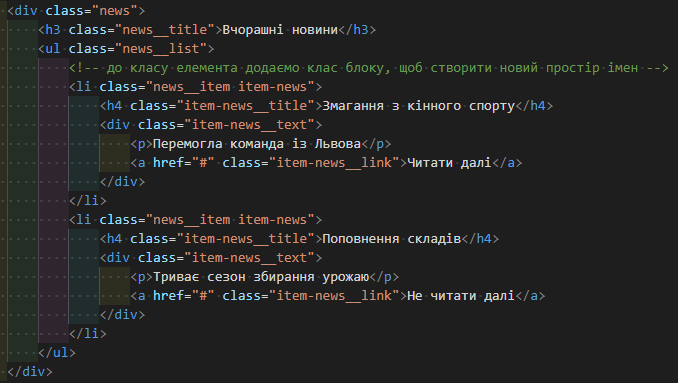
Навігація (складний варіант)



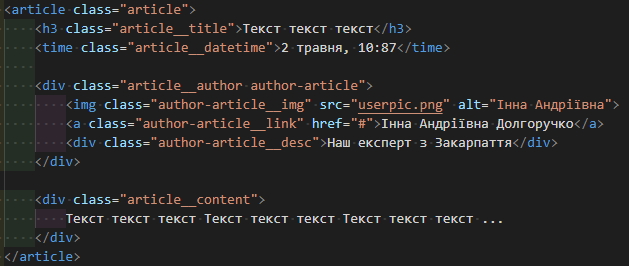
Віджет у бічній колонці



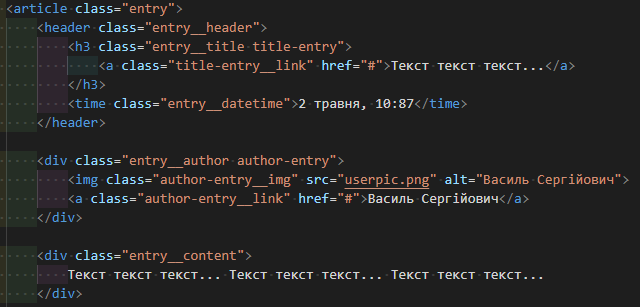
Блок новин



Стаття чи пост у блозі (простий варіант)



Стаття чи пост у блозі (складний варіант)





Розмітка сторінки

