МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНЙ УНІВЕРСИТЕТ

Кафедра програмного забезпечення автоматизованих систем

РЕФЕРАТ

На тему «Якість коду та розрозбки ПЗ»

з дисципліни "Програмування WEB Додатків"

Перевірив роботу: Виконав роботу:

"\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ студент гр. ПЗС-2244

викладач\_\_\_\_\_\_\_\_\_\_\_\_\_ Брус В.М.

Черкаси 2022

[ВСТУП 3](#_Toc122019473)

[1 ЗАГАЛЬНА ІНФОРМАЦІЯ 4](#_Toc122019474)

[2 СТИЛІЗАЦІЯ В МОВАХ ПРОГРАМУВАННЯ 6](#_Toc122019475)

[2.1 HTML 6](#_Toc122019476)

[2.2 CSS 10](#_Toc122019477)

[2.3 JAVASCRIPT 12](#_Toc122019478)

[КОНТРОЛЬНІ ПИТАННЯ 14](#_Toc122019479)

[ВИКОРИСТАНА ЛІТЕРАТУРА 16](#_Toc122019480)

[ВИСНОВКИ 17](#_Toc122019481)

# ВСТУП

Якість коду може визначатися різними критеріями. Деякі з них мають значення лише з погляду людини. Наприклад, те, як відформатовано текст програми, зовсім не є важливим для комп'ютера, але може мати серйозне значення для подальшої розробки. Багато стандартів оформлення коду, що визначають специфічні для використовуваної мови угоди і задають ряд правил, що покращують читаність коду, мають на меті полегшити майбутню розробку ПЗ, що включає налагодження та оновлення. Існують і інші критерії, що визначають, чи «добре» написаний код, наприклад, такі, як структурованість - ступінь логічного розбиття коду на ряд керованих блоків.

Якість програмного забезпечення - характеристика програмного забезпечення як ступеня його відповідності вимогам. При цьому вимоги можуть трактуватися досить широко, що породжує низку незалежних визначень поняття.

# 1 ЗАГАЛЬНА ІНФОРМАЦІЯ

Фактор якості ПЗ — це нефункціональна вимога до програми, яка зазвичай не описується в договорі із замовником, проте є бажаною вимогою, що підвищує якість програми.

Деякі з факторів якості:

* Зрозумілість - Призначення ПЗ має бути зрозумілим, із самої програми та документації.
* Повнота - Усі необхідні частини програми мають бути представлені та повністю реалізовані.
* Стислість - Відсутність зайвої інформації, що дублюється. Частини коду, що повторюються, повинні бути перетворені на виклик загальної процедури. Те саме стосується й документації.
* Портованість - Легкість в адаптації програми до іншого оточення: іншої архітектури, платформи, операційної системи чи її версії.
* Узгодженість - По всій програмі та в документації повинні використовуватися ті самі угоди, формати та позначення.
* Зручність використання - простота та зручність використання програми. Ця вимога стосується насамперед інтерфейсу користувача.
* Надійність - відсутність відмов та збоїв у роботі програм, а також простота виправлення дефектів та помилок:
* Структурованість
* Ефективність - наскільки раціонально програма належить до ресурсів (пам'ять, процесор) у виконанні своїх завдань.
* Безпека

Крім технічного погляду на якість ПЗ, існує оцінка якості з позиції користувача. Для цього аспекту якості іноді використовують термін "юзабіліті". Досить важко отримати оцінку юзабіліті для заданого програмного продукту.

Найбільш важливі питання, що впливають на оцінку:

* Чи є інтерфейс користувача інтуїтивно зрозумілим?
* Наскільки просто виконувати прості, часті операції?
* Наскільки легко виконуються складні операції?
* Чи видає програма зрозумілі повідомлення про помилки?
* Чи завжди поводиться так як очікується?
* Чи є документація та наскільки вона повна?
* Чи завжди затримки з відповіддю програми є прийнятними?

# СТИЛІЗАЦІЯ В МОВАХ ПРОГРАМУВАННЯ

## 2.1 HTML

Протокол:

Не вказуйте протокол під час увімкнення ресурсів на сторінку.

Опускайте назву протоколу (http:, https:) у посиланнях на зображення або інші медіа-ресурси, файли стилів або скрипти, звичайно, якщо ці файли доступні по обох протоколах.

Відсутність протоколу робить посилання відносним, що запобігає змішуванню ресурсів із різних протоколів і трохи зменшує розмір файлів.

Відступи:

Завжди використовуйте для відступу два пробіли. Не використовуйте табуляцію та не змішуйте табуляцію з пробілами.

Реєстр:

Завжди пишіть у нижньому регістрі. Весь код повинен бути написаний у нижньому регістрі: Це стосується назв елементів, назв атрибутів, значень атрибутів (крім тексту/CDATA), селекторів, властивостей та їх значень (крім тексту).

Кодування:

Використовуйте UTF-8 (без BOM). Переконайтеся, що редактор використовує кодування UTF-8 без мітки порядку байтів (BOM).

Вкажіть кодування в HTML шаблонах та документах за допомогою <meta charset="utf-8">. Опускайте кодування для сss-файлів: для них UTF-8 встановлено за замовчуванням.

Коментарі:

По можливості пояснюйте свій код, де це потрібно.

Використовуйте коментарі, щоб пояснити свій код: що він робить, за що відповідає і чому використовується обране рішення.

(Цей пункт не є обов'язковим, тому що немає сенсу очікувати, що код завжди буде добре задокументований. Корисність коментування залежить від складності проекту і може відрізнятися для HTML та CSS коду. )

Завдання:

Визначте завдання для переліку справ за допомогою TODO.

Визначте завдання за допомогою ключового слова TODO. не використовуйте інші формати, що часто зустрічаються, такі як @@.

Покладайте контакти (ім'я користувача або список адресатів) у круглі дужки: TODO (контакт).

Описуйте завдання після двокрапки, наприклад: TODO: Завдання.

Тип документа:

Використовуйте HTML5. HTML5 (HTML синтаксис) рекомендується для всіх html-документів: <!DOCTYPE html>.

Валідність HTML:

По можливості використовуйте валідний HTML.

Використовуйте валідний HTML-код, крім випадків, коли використання не дозволяє досягти розміру файлу, необхідного для потрібного рівня продуктивності.

Використовуйте такі інструменти як W3C HTML validator, щоб перевірити валідність коду.

Валідність - це важлива і при цьому вимірювана якість коду. Написання валідного HTML сприяє вивченню технічних вимог та обмежень та забезпечує правильне використання HTML.

Семантика:

Використовуйте HTML так, як це було задумано.

Використовуйте елементи (Іноді неправильно звані “тегами”) за призначенням: заголовки для заголовків, p для абзаців, a для посилань тощо.

Це полегшує читання, редагування та підтримку коду.

Альтернатива для мультимедіа:

Завжди вказуйте альтернативний вміст для мультимедіа.

Спробуйте вказати альтернативний вміст для мультимедіа: наприклад, для картинок, відео або анімацій, заданих за допомогою canvas. Для картинок це осмислений альтернативний текст (alt), а для відео та аудіо розшифровки тексту та підпису якщо це можливо.

Альтернативний вміст може допомогти людям з обмеженими можливостями. Наприклад, людині зі слабким зором складно зрозуміти, що на картинці якщо для неї не заданий @alt. Іншим людям може бути важко зрозуміти про що йдеться у відео або аудіо запису.

Поділ відповідальності:

Розділяйте структуру, оформлення та скрипти. Тримайте структуру, оформлення та скрипти окремо та постарайтеся звести взаємодію між ними до мінімуму.

Переконайтеся, що документи та шаблони містять лише HTML, і що HTML використовується лише для завдання структури документа. Весь код, який відповідає за оформлення, перенесіть у файли стилів, а код, що відповідає за поведінку, — у скрипти.

Намагайтеся скоротити їх перетин до мінімуму, включаючи в шаблони мінімальну кількість файлів стилів і скриптів.

Відділення структури від подання та поведінки допомагає полегшити підтримку коду. Зміна шаблонів і документів HTML завжди займає більше часу, ніж зміна файлів стилів або скриптів.

Посилання-немоніки:

Не використовуйте посилання-немоніки.

Немає сенсу використовувати посилання-немоніки, такі як &mdash;, &rdquo;, або &#x263a;, коли всі команди у файлах, редакторах використовують одне кодування (UTF-8)

Єдиний виняток із цього правила - службові символи HTML (наприклад < і &) а також допоміжні та "невидимі" символи.

Необов'язкові теги:

Для зменшення розміру файлів і кращого читання коду можна опускати необов'язкові теги. У специфікації HTML5 є список необов'язкових тегів.

(Може знадобитися деякий час для того, щоб цей підхід почав використовуватися повсюдно, тому що це сильно відрізняється від того, чому зазвичай вчать веб-розробників. З точки зору, узгодженості, і простоти коду найкраще опускати всі необов'язкові теги, а не деякі з них).

Атрибут 'type':

Не вказуйте атрибут типу під час підключення стилів та скриптів до документа.

Не використовуйте атрибут type при підключенні стилів (крім варіантів, коли використовується щось крім CSS) і скриптів (крім варіантів, коли це не JavaScript).

Вказувати атрибут type в даному випадку не обов'язково, тому що HTML5 використовує text/css і text/javascript за замовчуванням. Це працюватиме навіть у старих браузерах.

Форматування:

Виділяйте новий рядок для кожного блокового, табличного або спискового елемента та ставте відступи для кожного дочірнього елемента.

Незалежно від стилів заданих для елемента (CSS дозволяє змінити поведінку елемента за допомогою властивості display) переносіть кожен блоковий або табличний елемент на новий рядок.

Також ставте відступи для всіх елементів, вкладених у блоковий або табличний елемент.

### 2.2 CSS

Валідність CSS:

По можливості використовуйте валідний CSS-код.

Крім випадків, де необхідний браузеро-залежний код або помилок валідатора, використовуйте валідний CSS код.

Використовуйте такі інструменти, як W3C CSS Валідатор для перевірки свого коду.

Валідність - це важлива і при цьому вимірювана якість коду. Написання валідного CSS допомагає позбутися надлишкового коду та забезпечує правильне використання таблиць стилів.

Ідентифікатори та назви класів:

Використовуйте шаблонні або імена класів та ідентифікатори.

Замість використання шифрів, або опису зовнішнього вигляду елемента, спробуйте в імені класу або ідентифікатора висловити сенс його створення, або дайте йому шаблонне ім'я.

рекомендується вибирати імена, що відображають сутність класу, тому що їх простіше зрозуміти і, швидше за все, не знадобиться міняти у майбутньому.

Шаблонні імена — це просто варіант назви для елементів, які не мають спеціального призначення або не відрізняються від своїх братів і сестер. Зазвичай вони необхідні як “Помічники”.

Використання функціональних або шаблонних імен зменшує необхідність непотрібних змін у документі або шаблоні.

Назви ідентифікаторів та класів:

Для ідентифікаторів і класів використовуйте такі довгі імена, наскільки потрібно, але настільки короткі, наскільки це можливо.

Спробуйте сформулювати, що саме повинен робити даний елемент, при цьому будьте стислі наскільки можливо.

Таке використання класів та ідентифікаторів робить свій внесок у полегшення розуміння та збільшення ефективності коду.

Селектори типу:

Уникайте використання імен класів або ідентифікаторів із селекторами типу елемента.

Крім випадків, коли це необхідно (наприклад з класами-помічниками), не використовуйте назви елементів з іменами класів або ідентифікаторами.

Скорочені форми запису властивостей:

Використовуйте скорочені форми запису властивостей, де це можливо.

CSS пропонує безліч різних скорочених форм запису (наприклад font), які рекомендується використовувати скрізь, де це можливо, навіть якщо задається лише одне зі значень.

Використання скороченого запису властивостей корисне для більшої ефективності та кращого розуміння коду.

Префікси:

Попереджайте селектори унікальними для поточної програми префіксами. (не обов'язково)

У великих проектах, а також у коді, який буде використовуватися для інших проектів або інших сайтах, використовуйте префікси (як простори імен) для ідентифікаторів та імен класів. Використовуйте короткі унікальні назви з наступним дефісом.

Використання просторів імен дозволяє запобігти конфліктам імен і може полегшити обслуговування сайту. Наприклад при пошуку та заміні.

Розділювачі у класах та ідентифікаторах:

Розділяйте слова в ідентифікаторах та іменах класів за допомогою дефісу.

Не використовуйте нічого, крім дефісу, для з'єднання слів та скорочень у селекторах, щоб підвищити зручність читання та легкість розуміння коду.

## 2.3 JAVASCRIPT

Фігурні дужки:

У більшості JavaScript проектів фігурні дужки пишуться в так званому «єгипетському» стилі з дужкою, що відкриває, на тому ж рядку, що і відповідне ключове слово – не на новому рядку. Перед відкриваючою дужкою має бути пробіл, як тут:

Довжина рядка:

Максимальну довжину рядка узгоджують у команді. Зазвичай це 80 чи 120 символів.

Відступи:

Горизонтальні відступи: 2 або 4 пробіли.

Горизонтальний відступ виконується за допомогою 2 або 4 пробілів або символу табуляції (клавіша Tab). Який із них вибрати – це вже на ваш розсуд.

Не повинно бути більше 9 рядків коду поспіль без вертикального відступу.

Крапка з комою:

Крапки з комою повинні бути присутніми після кожного виразу, навіть якщо їх, здавалося б, можна пропустити.

Є мови, у яких крапка з комою необов'язкова та рідко використовується. Однак у JavaScript трапляються випадки, коли перенесення рядка не інтерпретується, як точка з комою, що може призвести до помилок.

Якщо ви – досвідчений розробник JavaScript, то можна вибрати стиль коду без крапок з комою, наприклад StandardJS. В іншому випадку, краще буде використовувати крапки з комою, щоб уникнути непорозумінь та помилок. Більшість розробників їх ставлять.

Рівні вкладеності:

Рівень вкладеності має бути небагато. Наприклад, у циклі буває корисно використовувати директиву continue, щоб уникнути зайвої вкладеності.

# КОНТРОЛЬНІ ПИТАННЯ

1. Що таке чистий код?

Чистий код написаний максимально лаконічно та елегантно, без дублювання. Він структурований таким чином, щоб було легко читати як людям, так і комп'ютерам. Кожен може написати код, зрозумілий комп'ютеру, але хороший програміст може написати код, зрозумілий людині.

1. В чому полягають основні особливості створення «якісного» ПЗ?

Якість програмного забезпечення - це те, наскільки програмне забезпечення задовольняє вимогам, що висуваються до нього. Висувні вимоги можуть залежати від багатьох критеріїв, що визначаються виходячи зі сфери застосування програмного продукту.

1. Що таке валідність коду?

Термін утворений він від англійської valid - відповідний вимогам, допустимий, прийнятний, правильний. Валідність коду - це вихідний код сторінки, який має бути написаний згідно з правилами веб-стандартів та мови розмітки, прописаними у стандарті W3C – організації.

1. Які переваги в створенні опимізованого коду для сайту?

Оптимізація необхідна для збільшення швидкості завантаження та роботи сайтів, що зрештою підвищує задоволеність користувачів.

З точки зору User Experience фронтенд повинен забезпечувати швидке завантаження та роботу веб-сторінок.

1. Що таке зрозумілий код?

Це код, який буде приємно читати, дізнаватися, обслуговувати, доповнювати, масштабувати іншому розробнику, а не тому, що його створював. Досягти цього реально.

1. Що таке «usability»?

Це зручність використання, придатність використання - здатність продукту бути зрозумілим, вивченим, використовуваним і привабливим для користувача в заданих умовах, властивість системи, продукту або послуги, за наявності якого конкретний користувач може експлуатувати систему у певних умовах для досягнення встановлених цілей з необхідною результативністю, ефективністю та задоволеністю.

1. Чому кодування UTF-8 використовують найбільш часто?

UTF-8 може закодувати будь-який Unicode символ, що дозволяє уникнути використання кодових сторінок або, наприклад, методів визначення того, який набір символів використовується в даний момент, а також дозволяє виводити кілька записів одночасно.

1. Що таке клас в програмуванні?

Клас - в об'єктно-орієнтованому програмуванні, модель для створення об'єктів певного типу, що описує їх і визначає алгоритми для роботи з цими об'єктами.

Іншими словами, клас є засобом для введення абстрактних типів даних у програмний проект. Інші описувачі абстрактних типів даних — метакласи, інтерфейси, структури, перерахування — характеризуються якимись своїми особливостями. Суть відмінності класів у тому, що з завдання типу даних, клас визначає одночасно як інтерфейс, і реалізацію всім своїх екземплярів (тобто. об'єктів), тому виклик метода-конструктора обов'язковий.

1. Для чого використовується атрибут type в HTML?

Атрибут типу визначає тип елемента введення. Тип за замовчуванням – text. Цей атрибут не є обов'язковим, проте найкраще завжди визначати його. Атрибут type для тега <input> підтримується майже всіма браузерами.

1. Для чого використовують коментарі в коді?

Для пояснення до вихідного тексту програми, що знаходяться безпосередньо всередині коду, що коментується. Синтаксис коментарів визначається мовою програмування. З погляду компілятора чи інтерпретатора, коментарі — частина тексту програми, яка впливає її семантику. Коментарі не впливають на результат компіляції програми чи її інтерпретацію. Крім вихідних текстів програм, коментарі також застосовуються у мовах розмітки та мовах опису.

# ВИКОРИСТАНА ЛІТЕРАТУРА

1. https://developers.google.com/style/html-formatting

2. https://google.github.io/styleguide/htmlcssguide.html

3. https://google.github.io/styleguide/jsguide.html

4. https://qalight.ua/baza-znaniy/yakist-programnogo-zabezpechennya/

5. <https://dou.ua/forums/topic/35450/>

6. <https://habr.com/>

7. https://www.wikipedia.org/

# ВИСНОВКИ

Коли ми думаємо про написання «кращого» коду, ми маємо поставити собі запитання: «Що зробить код більш читабельним та легким для розуміння?» та «Що може допомогти уникнути помилок?». Це основні моменти, про які слід пам'ятати при виборі та обговоренні стилів коду.

Якість пз - це набір властивостей продукту, що характеризують його здатність задовольнити встановлені або передбачувані потреби замовника