ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ
ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка адаптивної платформи на JavaScript з алгоритмами AI для аналізу даних онлайн-ігор та рекомендацій»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Микита Биков
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСД-42

Микита Биков
Ім'я, ПРІЗВИЩЕ

Керівник: Шахматов І.О.
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент:
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти бакалавр
Спеціальність Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС
_____Каміла СТОРЧАК
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Биков Микита Роланович
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Розробка адаптивної платформи на JavaScript з алгоритмами AI для аналізу даних онлайн-ігор та рекомендацій

керівник кваліфікаційної роботи Шахматов Іван Олександрович
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
1. Науково-технічна література з теми адаптивних платформ на JavaScript.
2. Основи штучного інтелекту та алгоритмів машинного навчання.
3. Принципи адаптивного дизайну для різних пристроїв та екранних розмірів.
4. Методології аналізу даних в онлайн-іграх.
5. Дослідження поведінки користувачів в онлайн-іграх та їх вподобань.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Принципи адаптивного дизайну для веб-застосунків
2. Технології розробки адаптивних платформ на JavaScript
3. Вибір та налаштування алгоритмів штучного інтелекту для аналізу даних онлайн-ігор
4. Аналіз даних гравців та поведінкові моделі в онлайн-іграх
5. Тестування та верифікація функціоналу адаптивної платформи

5. Ілюстративний матеріал: *презентація*

6. Дата видачі завдання: «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Розробка презентації застосунку	01.05.24	
2	Підбір науково-технічної літератури	05.05.24	
3	Дослідження програмних засобів	09.05.24	
4	Моделювання об’єкту проектування	12.05.24	
5	Розробка функціоналу API	14.05.24	
6	Вступ, висновки, реферат	20.05.24	

Здобувач(ка) вищої освіти _____ Микита Биков
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Іван Шахматов
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Розробка адаптивної платформи на JavaScript з алгоритмами AI для аналізу даних онлайн-ігор та рекомендацій»: 69 с., 58 мал., 5 інформаційних джерел.

Об’єкт розробки – адаптивна платформа для знаходження преметів в онлайн грі Warframe

Мета проекту – отримати сайт – знаходження преметів в онлайн грі Warframe.

Методи розробки – мова програмування HTML,CSS,JAVASCRIPT, інтегроване середовище розробки Visual Studio Code**,** HTTP-інтерфейс WarframeAPI.

МЕТА-ТЕГИ, РОЗРОБКА, БАЗА ДАНИХ.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	Бази даних
URL *(Uniform Resource Locator)*	Визначник місцезнаходження сайту в мережі Інтернет
PoC «Proof of concept»	Прототип для перевірки життєздатності ідеї або концепції
API (Application programming interface)	Програмний інтерфейс
HTTP *(HyperText Transfer Protocol)*	Протокол передачі даних, що використовується в комп'ютерних мережах
AI (Artificial Intelligence)	Штучний інтелект
WWW (World Wide Web)	Розподілена система, що надає доступ до пов'язаних між собоюдокументів
CSS (Cascading Style Sheets)	Каскадні таблиці стилів
JS (JavaScript)	Мультипарадигменна мова програмування.
ОС (Operating System)	Операційна система
PHP (Hypertext Preprocessor)	Скриптова мова програмування,призначена для розробки веб- застосунків
XML (EXtensible Markup Language)	Мова розмітки, що нагадує HTML.
DOM (Document Object Model)	Представляє структуру документа

ЗМІСТ

ВСТУП

Онлайн-ігри привернули увагу мільйонів гравців у всьому світі, та з кожним роком кількість гравців тільки росте. Хто вони ці гравці? Якого вони віку? Який їх соціальний стан? Що їх цікавить? Що саме робить гру такою привабливою та захоплюючою для них?

У повсякденному житті ми переживаємо стрес та маємо потребу у відпочинку. Кожен робить свій вибір як відпочивати, це може бути: дивитися Netflix, читати книги чи гра в онлайн-ігри. За допомогою онлайн-ігор гравець поринає в іншу реальність та отримує нові навички та відчуття, та сприяють розвитку пізнавальних, соціальних та емоційних якостей, наприклад:

- Покращення навичок вирішення проблем - проведення складних квестів та головоломок у онлайн іграх стимулює критичне мислення та навички вирішення проблем. Гравцям часто потрібно розробляти творчі стратегії для подолання труднощів, що сприяє аналітичному мисленню, яке можна застосувати у реальних життєвих ситуаціях.
- Розвиток навичок - ігри вимагають володіння конкретними навичками, такими як координація рухів, стратегічне планування та управління ресурсами. Регулярна практика може призвести до значного покращення навичок, які можуть бути корисними за межами геймінгу.
- Зняття стресу та розваги - поглиблення у захоплюючий віртуальний світ може стати втечею від щоденних тиску, сприяючи релаксації та задоволенню.
- Культурні та освітні аспекти - ігри включають історичні, культурні та освітні елементи. Гравці можуть здобувати уявлення про різні історичні періоди, цивілізації та концепції, роблячи навчання захоплюючим та інтерактивним досвідом.

Так багато цікавої інформації з гри, чи не правда? Проаналізувавши її ми можемо вплинути з однієї сторони на розвиток та покращення гри що

сподобаються гравцям та мати вплив підвищення прибутку від гри що є цікавим для розробників гри.

Метою дипломного проекту є розробка адаптивної платформи - інтерактивний веб-застосунок створений за допомогою мови програмування JavaScript, який може адаптуватися до різних пристроїв і екранних розмірів. Цей застосунок буде використовувати розумні алгоритми штучного інтелекту для аналізу даних, зібраних з онлайн-ігор, і надавати рекомендації користувачам щодо їхньої гри або вибору контенту а розробник отримає зворотний зв'язок для покращення гри.

Чому це важливо?

- Зручність для користувачів: Адаптивна платформа забезпечує зручний доступ до інформації та функціоналу незалежно від пристрою, який використовує користувач.
- Покращення користувацького досвіду –використання алгоритмів штучного інтелекту дозволяє платформі аналізувати поведінку користувачів та надавати персоналізовані рекомендації, що покращує їхній досвід використання.
- Аналітика та оптимізація - платформа може збирати та аналізувати дані про користувачів та їхню взаємодію з онлайн-іграми, що дозволяє розробникам оптимізувати функціонал платформи та покращувати групові рекомендації.
- Фінанси – гра це не тільки розвага, це і одночасно бізнес, аналіз вподобань користувача при здійсненні покупок у грі дозволить додавати у гру нові цікаві елементи та заробляти на цьому гроші.

Отже, розробка такої платформи може призвести до покращення користувацького досвіду в онлайн-іграх, забезпечуючи зручний доступ до контенту та персоналізовані рекомендації незалежно від пристрою:

– На комп'ютері - користувач може відкрити сайт на своєму комп'ютері з великим екраном. Адаптивний дизайн забезпечить оптимальне

розміщення елементів інтерфейсу, так щоб користувач міг легко переглядати контент, використовуючи мишу або клавіатуру.

- На планшеті - користувач відкриє сайт на планшеті, адаптивний дизайн автоматично перестроїть інтерфейс, забезпечивши оптимальне використання доступного екранного простору. Це може включати зміну розміру кнопок, оптимізацію розташування елементів та інше, щоб полегшити навігацію по сайту на планшеті.
- На смартфоні - користувач відкриє сайт на смартфоні, адаптивний дизайн буде оптимізований для використання на маленькому екрані. Це може означати перетворення меню в меню з об'ємним списком, розташування кнопок внизу екрану для зручного використання однією рукою та інші зміни, щоб полегшити навігацію на мобільному пристрої.

Отже, використання адаптивного дизайну дозволяє платформі автоматично адаптуватися до різних пристроїв, забезпечуючи зручний доступ до інформації та функціоналу для користувачів незалежно від їхнього пристрою.

Виходячи з поставленої мети, сформовано наступні завдання:

- проаналізувати обрану предметну область;
- обрати технології та середовище для розробки;
- підготувати контент для наповнення сайта адаптивної;
- розробити схему інтерфейс сайту для адаптивної моделі для ноутбуку;
- Обрати модель штучного інтелекту яка буде давати рекомендації
- Виконано демонстрацію PoC «Proof of concept» працездатності адаптивну.

Проект має велику цінність як для користувача так і для розробника: зручний і зрозумілий контент, корисні рекомендації та адаптований інтерфейс.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Warframe як метод дослідження

Warframe — кооперативна відеогра-шутер від третьої особи, розроблена канадською студією Digital Extremes.

Уперше випущена в 2013 році, гра з тих пір отримала багато оновлень і доповнень, які розширюють її вміст і геймплей.

У Warframe гравці беруть на себе роль тенно, стародавньої раси воїнів, яка через століття прокидається від кріосну та стає втягненою у велику космічну війну між різними фракціями.

Tenno використовує спеціальні костюми під назвою Warframes, кожен з яких має унікальні здібності та може бути налаштований за допомогою різноманітної зброї та модів.

Ключові особливості гри: Спільна гра: Гравці можуть об’єднуватися в групи до чотирьох осіб для виконання місій на різних планетах і місцях Сонячної системи.

Різноманітність місій : Місії включають різні типи місій, такі як атака бази, оборона точки, викрадення, виживання з хвиль ворогів тощо.

Широкий вибір зброї та варфреймів: гравці можуть збирати та налаштовувати різноманітні варфрейми та сотні видів зброї.

Прогрес і модифікація: виконуючи місії та завдання, гравці отримують ресурси, які можна використовувати для покращення свого Warframe, зброї та здібностей.

Динамічний світ : Ігровий світ регулярно оновлюється новими подіями, місіями та розширеннями, щоб зацікавити гравців.

Warframe відомий широкими можливостями налаштування, великою базою знань і активною спільнотою, яка продовжує зростати завдяки регулярним оновленням від розробників.

Геймплей: Боротьба та співпраця: Ігри відбуваються в реальному часі та зазвичай складаються з боротьби з ворогами з різних фракцій у великих і

деталізованих місцях.

Гравці можуть об’єднуватися та разом виконувати місії, використовуючи різні стратегії та комбінації навичок.

Warframes: Серцем гри є Warframes, або броньовані екзоскелети, які дають гравцям унікальні здібності та характеристики.

Кожен Warframe має унікальні здібності, які підходять для різних стилів гри.

Зброя: Гравці також можуть використовувати різноманітну зброю, включаючи пістолети, мечі, кинджали та сокири.

Кожен тип зброї має унікальні характеристики та можливості модифікації.

Місії та вміст: Warframe пропонує широкий вибір типів місій, від класичних бойових завдань до захоплюючих квестів зі складними сюжетними лініями.

Крім того, гра регулярно оновлюється новим контентом, таким як місії, Warframe, зброя та прикраси.

Прогрес: Оновлення та налаштування: Гравці можуть використовувати отримані ресурси для покращення свого Warframe та зброї, відкриваючи нові функції та модифікації.

Це дозволяє створювати індивідуальні бійці, які відповідають вашим особистим смакам і стилю гри.

Майстерність: Гравці заробляють Майстерність, виконуючи різноманітні завдання та досягаючи цілей у грі.

Вищі рівні майстерності відкривають нову зброю, доступ до Warframes та інші переваги.

Соціальні аспекти: Спільнота: Warframe має велику й активну спільноту гравців, які часто спілкуються через форуми, соціальні мережі та групи месенджерів.

Взаємодія з іншими гравцями – це дуже весело, а також може бути стратегічною можливістю.

Торгівля та ринки: У грі є внутрішня економіка, де гравці можуть обмінюватися різними предметами, ресурсами та послугами.

Історія та світ: Унікальний всесвіт: Warframe розгортається у всесвіті наукової фантастики, де людство розкидано по зоряних системах, а різні фракції змагаються

за владу та ресурси.

Глибока історія: Хоча гра зосереджена в основному на дії, є також багата історія та традиції, які гравці можуть досліджувати за допомогою квестів і взаємодії персонажів.

Графіка та звук: Вражаюча візуальна атмосфера: Warframe характеризується вражаючою графікою та детальними локаціями, які роблять ігровий світ відчутним та яскравим.

Ефективний звуковий дизайн: Звуковий дизайн гри доповнює графіку та створює насичену, захоплюючу атмосферу для бою та дослідження.

Регулярні оновлення: Регулярні виправлення та доповнення: Розробники Warframe постійно додають новий вміст і покращують гру, щоб підтримувати інтерес і мотивацію гравців продовжувати гру.

Події та спеціальні події: Окрім основного вмісту, у грі регулярно проводяться тимчасові події та рекламні акції, які пропонують унікальні завдання та нагороди.

Кросплатформна гра: Кросплатформна гра: Warframe підтримує кросплатформну гру.

Це означає, що гравці з різних платформ можуть грати разом.

Портативність прогресу: Гравці можуть зберігати свій прогрес і вміст гри під час переходу з однієї платформи на іншу, а досягнення та нагороди можна зберігати на різних пристроях.

2 Створення першого в історії вебсайту

Тім Бернерс-Лі, фізик з CERN у Женеві, створив перший вебсайт.

У березні 1989 року він запропонував ідею використання гіпертексту для передачі даних через глобальну мережу Інтернету.

Бернерс-Лі створив свій перший вебсайт з адресою info.cern.ch ще в 1990 році.

Цей сайт містив детальний опис нової технології WWW (World Wide Web), яка базується на принципах URL-адресації в Інтернеті, протоколі передачі інформації HTTP і мові розмітки гіпертексту HTML.

Бернерс-Лі розробив Enquire у 1980 році, до створення першого вебсайту.

Цей проект мав на меті створити єдиний інформаційний простір , який би інтегрував Інтернет, комп'ютери користувачів і гіпертекст.

Це можливо завдяки принципу випадкових асоціацій.

Enquire використовував гіпертекст , щоб забезпечити зручний доступ до інформації на веб-сторінках і зробити посилання на ці сторінки через посилання.

Бернерс-Лі використав цей підхід, щоб продемонструвати переваги інновацій колегам із CERN.

Вчений не лише представив концепцію першого сайту , а й використав попередні дослідження, особливо напрацювання Enquire, для практичної реалізації інновацій у веб-середовищі.

У той час info.cern.ch вважався першим веб-сервером на основі комп'ютера NeXT.

Цей сервер розміщений у лабораторії CERN і визнаний першим у світі вебсайтом.

Перші веб-сторінки містили інформацію про проект , перший в історії веб-браузер WWW, технічні деталі для створення інших веб-сторінок і гіпертекстові дані.

Також ми детально розповіли, як шукати інформацію в Інтернеті, принципи встановлення браузера, сервер та порядок роботи браузера.

Цей сайт став повноцінним Інтернет-каталогом, і Тім Бернерс-Лі пізніше опублікував список посилань на нові вебсайти з нього.

З 1991 року веб-сервери почали поширюватися на інші європейські установи, і незабаром до них приєдналися американські колеги, які скористалися науковими розробками.

Сервер з'явився в Stanford Linear Accelerator Center SLAC.

Наприкінці 1992 року у світі було відомих лише 26 веб-серверів, але до жовтня 1993 року це число зросло до 200.

Цікаво відзначити, що ПК NeXT, над яким працював Тім Бернерс-Лі, був представлений у CERN на виставці Microcosm як перший у світі веб-сервер, веб-редактор і гіпермедійний браузер.

Стандарт WWW був визнаний і затверджений у травні 1991 року в CERN у Женеві, а Тім Бернерс-Лі був визнаний батьком основних веб-технологій (URI/URL, HTTP, HTML).

Тім Бернерс-Лі зараз очолює Консорціум World Wide Web (WWW Consortium), який він заснував.

Наразі перед компанією стоїть завдання з розробки та впровадження нових стандартів.

3 Основні елементи мови HTML та їх ключові характеристики

3.1 Еволюція мови розмітки HTML

Мова гіпертекстової розмітки HTML (HyperText Markup Language) була розроблена Тімом Бернерсом-Лі в Європейській організації ядерних досліджень (CERN) на початку 1990-х років.

HTML базується на SGML (стандартна узагальнена мова розмітки).

Основною ідеєю SGML було створення незалежної від платформи мови для розмітки документів.

Ця ідея була реалізована командою під керівництвом Чарльза Голдфарба з IBM, який вирішив відокремити структуру документа від дизайну .

В основу розмітки було засноване на введенні в код спеціальних керуючих структур, які визначали елементи структури без надання інструкцій для візуального відображення.

Макет документа зберігається в окремому файлі, який називається таблицею стилів (CSS)" для уточнення.

Ця структура дозволяє відображати ту саму інформацію з різними візуальними параметрами.

Розробники системи дійшли висновку, що документи можуть бути надійно оброблені лише за умови відповідності єдиному стандарту, а документи, які не відповідають цьому стандарту, повинні бути відхилені.

З цією метою було запроваджено механізм визначення типу документа (DTD) .

DTD, як і таблиця стилів, є зовнішнім файлом, пов’язаним із відповідним документом.

3.2 Основні елементи HTML

<DOCTYPE>

Цей тег призначений для визначення типу поточного документа (DTD – Document Type Definition).
Це необхідно для того, щоб браузер розумів, як правильно інтерпретувати **веб-сторінки.**
Існують різні версії HTML, у **тому числі** XHTML (EXtensible HyperText Markup **Language),** яка **схожа на** HTML, але має **інший** синтаксис.
Додавши тег до першого рядка коду, браузер може чітко визначити, за яким стандартом відображати веб-сторінку, **таким чином уникаючи** непорозумінь і **відображаючи вміст правильно.**

<html lang="ua">

Атрибут lang використовується для визначення мови елемента.
Тобто мова, якою написані нередаговані елементи, або мова, якою користувачі редагують редаговані елементи.

<head>

Теги HTML містять елементи, які допомагають браузерам маніпулювати даними.
Крім того, мета-теги можна розмістити всередині контейнера, щоб служити сховищем інформації для браузерів і пошукових систем.
Такі мета-теги можуть містити заголовки документів, сценарії, стилі та інші параметри.

<title>

Тег HTML встановлює заголовок документа так, як він відображається у вікні браузера або на вкладці сторінки.
Вміст елемента містить лише текст, усі теги всередині елемента ігноруються.

<body>

Теги HTML використовуються для зберігання вмісту веб-сторінки, яка відображається у вікні браузера.
Весь основний вміст і скрипти повинні бути розміщені в тегах.
Ця інформація включає текст, зображення, теги, таблиці та сценарії.

Цей тег використовується для встановлення позиції елемента на сторінці, кольору тексту, відступу, загальносторінкового шрифту тощо.

Однак набагато краще і зручніше використовувати стилі CSS, застосовані до селектора тіла.

Тим більше, що в різних браузерах вже підтримується більше тегів. (Рис. 3.2.1)

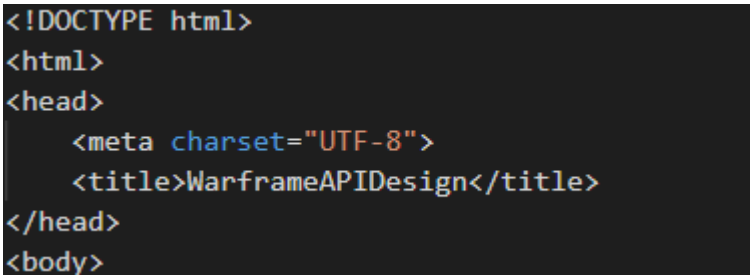


Рисунок 3.2.1 – Приклад структури сайту

3.3 Метадані документа

Метадані містять інформацію про сторінку, таку як стилі, сценарії та дані, які допомагають програмному забезпеченню (пошукові системи, браузери тощо) використовувати та відображати сторінку.
Інформацію про стиль і сценарій можна визначити на сторінці або отримати посилання з окремого файлу, що містить необхідні дані.

<base>

Тег HTML встановлює базову адресу (URL) для всіх відносних адрес (URL) у документі.
У документі допускається лише один елемент.

<link>

Теги визначають зв'язки між поточним документом і зовнішніми ресурсами.
Він часто використовується для посилань на таблиці стилів і створення піктограм сайтів, зокрема піктограм у стилі favicon, головних екранів і мобільних програм.

<meta>

Елементи в HTML використовуються для передачі різноманітної метаінформації про документ, такі як <base>, <link>, <script>, <style>, чи <title>. яка не відображається на самій сторінці, але є важливою для браузерів, пошукових систем та інших служб, які обробляють веб-сторінку.

3.4 Гіперпосилання

Гіперпосилання є важливою частиною роботи Інтернету та визначають характер його мережі.
У цій статті наведено синтаксис для створення посилань і описано найкращі методи їх використання.

Гіперпосилання є однією з найважливіших інновацій у світі Інтернету та визначають унікальну функціональність Інтернету.
Вони з самого початку визначили інтерактивність Інтернету, зробивши його справжньою «мережею».
Гіперпосилання дозволяють зв’язувати документи з іншими документами чи ресурсами на ваш вибір.
Ці посилання також надають можливість зв’язувати певні частини документа та робити веб-програму доступною за простою веб-адресою.
Це відрізняє веб-програми від традиційних локальних програм.

Створення гіперпосилань в Інтернеті вимагає використання певного синтаксису. У коді HTML це може виглядати наступним чином (Рис. 3.4.1)

```
<a href="посилання_на_ресурс">Текст або зображення, які стануть посиланням</a>
```

Рисунок 3.4.1 – Приклад гіперпосилання

Тут href визначає адресу (URL) ресурсу, на який **потрібно посилатися.**

Текст або зображення **в** тегу **діє як область активного** посилання.

Гіперпосилання **представляють** важливий аспект взаємодії в Інтернеті, **дозволяючи** ефективний обмін інформацією та зручний доступ до різноманітних ресурсів.

3.5 Базові теги

Теги HTML взяті в кутові дужки.
Слеші також використовуються в паперах.
Нижче наведено деякі теги, які використовуються частіше за інші.

- Теги <h1>-<h6> для заголовків
- Тег для визначення абзаців
- Тег <img> для вставки зображень
- Тег <a> для вставки посилань

Теги <h1> до <h6> використовуються для створення заголовків.
HTML має шість рівнів заголовків, від до .
використовується для позначення найважливіших заголовків і використовується для найменш важливих заголовків. (Рис. 3.5.1)

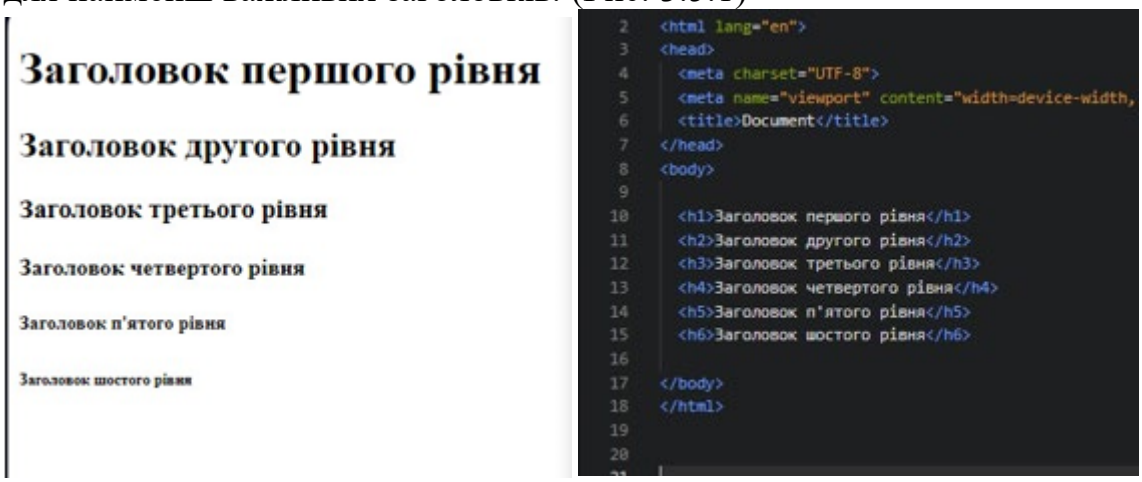


Рисунок 3.5.1 – Приклад заголовків

Парний тег використовується для розділення абзаців. За замовчуваннямбраузери автоматично додають відступи до і після . (Рис. 3.5.2)

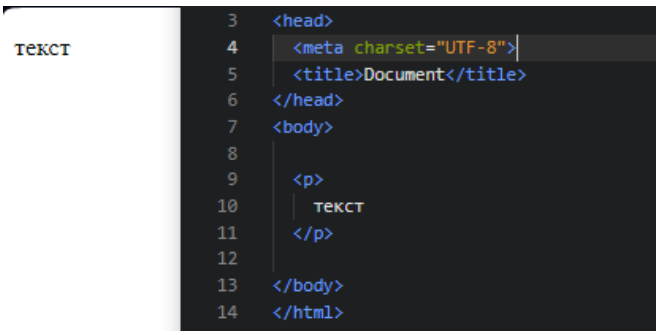


Рисунок 3.5.2 – Приклад абзацу

3.6 Форматування тексту

Вигляд тексту на екрані залежить від **тегів HTML, які використовуються** для **форматування тексту.**
Теги форматування поділяються на дві **категорії.**
теги фізичної розмітки, які відповідають за **стиль тексту (жирний,** курсив, шрифт **тощо),** і **логічні** теги розмітки, які **відповідають за семантичне** навантаження (наприклад, **те, що вам повідомляють пошукові системи).**

Теги HTML <b> і використовуються для визначення жирних шрифтів.
Різниця полягає в тому, що тег <b> є фізичним тегом розмітки, призначеним для виділення тексту без підкреслення сенсу.
Тег , навпаки, текст тепер вважається надзвичайно важливим
Вміст тегу дуже важливий для пошукових систем, і пристрої, які використовують відтворення тексту, можуть позначати тег спеціальною інтонацією. (Рис. 3.6.1)

Гра**Warframe** є вільним доступом із видом від
третьої особи
в жанрі **стрілянки** для одноосібного чи спільного грання

```
<title>Document</title>
</head>
<body>
<p>Гра<b>Warframe</b> є вільним доступом із видом від<br>третьої особи<br>
в жанрі <strong>стрілянки</strong>
 для одноосібного чи спільного грання </p>
</body>
</html>
```

Рисунок 3.6.1 – Оформлення за допомогою <b> і

Теги <i> та використовуються для визначення курсивного зображення шрифту.

Тег <i> є елементами фізичної розмітки, що означає, що вбудований текст відрізняється лише візуально та не має особливого семантичного значення для браузерів чи пошукових систем.

Тег , навпаки, допомагають виразно підкреслити фрагменти тексту та надати їм емоційного акценту. (Рис. 3.6.2)

Гра *Warframe* є вільним доступом із видом від
третьої особи
в жанрі *стрілянки* для одноосібного чи спільного грання

```
<body>
<p>Гра<i>Warframe</i> є вільним доступом із видом від<br>третьої особи<br>
в жанрі <em>стрілянки</em>
 для одноосібного чи спільного грання </p>
</body>
</html>
```

Рисунок 3.6.2 – Оформлення за допомогою <i> та

Тег <pre> використовується для вставки форматованого тексту в документ HTML. Текст, що міститься в цьому тегу, зберігає всі пробіли та розриви рядків і відображається браузером без об'єднання послідовних пробілів, як це відбувається за замовчуванням. (Рис. 3.6.3)

```
Гра Warframe є вільним доступом
 із видом від третьої особи в жанрі
 стрілянки для одноосібного чи спільного грання
```

```
<body>
 <pre>Гра Warframe є вільним доступом
 із видом від третьої особи в жанрі
 стрілянки для одноосібного чи спільного грання </pre>
</body>
</html>
```

Рисунок 3.6.3 – Оформлення за допомогою <pre>

Тег <small> встановлює розмір шрифту тексту на один розмір менший, ніж у батьківського елемента.
У HTML5 цей тег також використовується для включення інформації про авторські права та позначення дрібних або допустимих шрифтів. (Рис. 3.6.4)

Гра Warframe є вільним доступом із видом від третьої особи в жанрі стрілянки для одноосібного чи спільного грання

```
<p>Гра Warframe є вільним доступом
із видом від третьої особи в жанрі
<small>стрілянки для одноосібного чи спільного грання</small></p>
</body>
```

Рисунок 3.6.4 – Оформлення за допомогою <small>

Тег <del> використовується для ідентифікації частин тексту, які були видалені з документа.
Тег <s> використовується для позначення тексту, який втратив актуальність або більше не є важливим у контексті. (Рис. 3.6.5)

Гра Warframe є ~~вільним~~ доступом із видом від третьої особи в жанрі стрілянки для ~~одноосібного~~ чи спільного грання

```
<p>Гра Warframe є <del>вільним</del> доступом
 із видом від третьої особи в жанрі
 стрілянки для <s>одноосібного</s> чи спільного грання</p>
</body>
```

Рисунок 3.6.5 – Оформлення за допомогою <del> та <s>

3.7 Зображення

Атрибут src є обов'язковим, оскільки він визначає шлях до зображення.
Значенням атрибута src може бути ім'я або URL-адреса файлу.
Для тегу <img> також потрібен атрибут alt.
Його значенням є описовий текст, який відображається в браузері перед завантаженням зображення.
Цей текст також з'явиться у вашому браузері, коли ви наведете курсор на зображення. (Рис. 3.7.1)


```
<img src="/img/fone/Warframe.com" alt="Warframe">
```

Рисунок 3.7.1 – Вставка зображення Warframe

3.8 Списки

Мова гіпертекстової розмітки підтримує три типи списків, кожен зі своїми тегами.

Маркований список містить ненумеровані елементи без певного порядку.

Елемент <ul> block використовується для створення маркованих списків.

Кожен елемент списку починається початковим тегом <li> і закінчується кінцевим тегом .(Рис. 3.8.1)

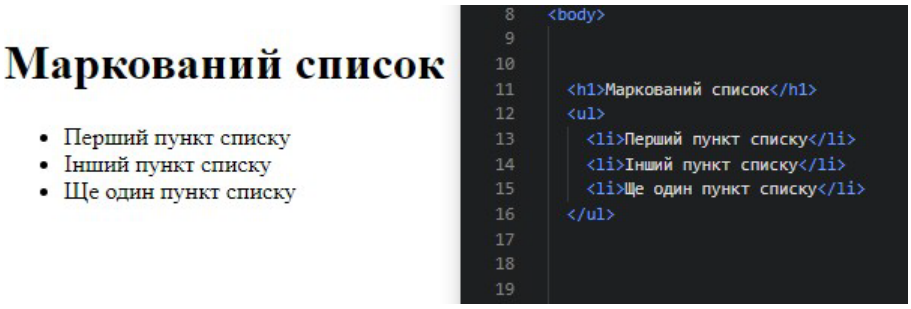


Рисунок 3.8.1 – Маркований список

Нумерований список містить елементи в певному порядку та використовує блоковий елемент <ol>.

Кожен елемент у пронумерованому списку починається тегом <li> (початок) і закінчується кінцевим тегом .

Елементам списку автоматично присвоюються номери. (Рис. 3.8.2)

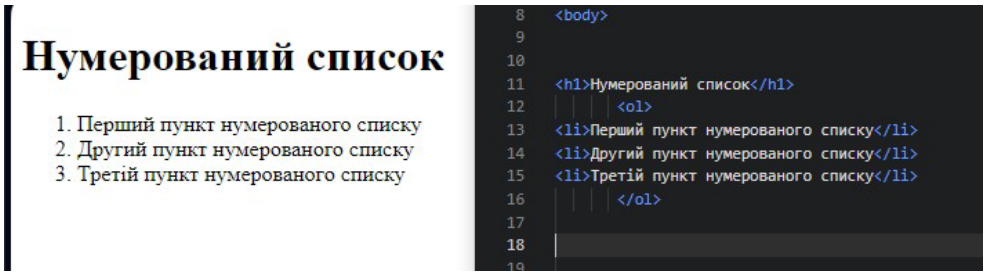


Рисунок 3.8.2 – Нумерований список

3.9 Таблиці HTML

Щоб створити таблицю в HTML, використовуйте елемент <table>.
Це контейнер для елементів, які визначають вміст таблиці.
Рядки таблиці визначаються за допомогою парних тегів блоків <tr>.
Кожен рядок таблиці записується окремо за допомогою тегу <tr>.
елементи в рядку або комірці визначаються тегом <td> (від англійського «табличні дані»).
Кожна клітинка представлена окремим тегом <td>, який містить вміст таблиці, наприклад числа та текст.
Заголовок рядка або стовпця таблиці визначається тегом <th> (від англійського «голова таблиці») і розміщується в першому рядку таблиці.
Браузер автоматично виділить його жирним шрифтом. (Рис. 3.9.1)

Місяць	**Дата**
Квітень	1.04.2024
Травень	1.05.2024

```
<table>
 <tr>
  <th>Місяць</th>
  <th>Дата</th>
 </tr>
 <tr>
  <td>Квітень</td>
  <td>1.04.2024</td>
 </tr>
 <tr>
  <td>Травень</td>
  <td>1.05.2024</td>
 </tr>
</table>
```

Рисунок 3.9.1 – Таблиця

3.10 Коментарі

Коментарі HTML використовуються для написання пояснень, нагадувань або додаткових пояснень щодо розділів коду HTML.

Це полегшує роботу з кодом, дозволяючи швидко знаходити потрібні частини та розуміти наміри програміста, який написав код (це корисно, якщо над сайтом працюють кілька людей (особливо важливо).

Коментарі також дозволяють тимчасово видаляти блоки коду, не видаляючи їх.

Якщо ви просто обернете цей сегмент тегами <!-- ... -->, він не відображатиметься у браузері. (Рис. 3.10.1)

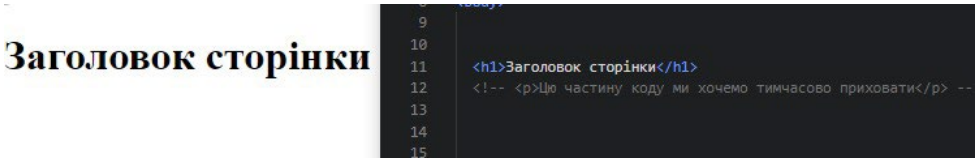


Рисунок 3.10.1 – Коментарі

4 CSS ЯК ІНСТРУМЕНТ ОФОРМЛЕННЯ

4.1 Введення в CSS

CSS (каскадні таблиці стилів) використовується для стилізації та компонування веб-сторінок.
CSS дозволяє змінювати шрифт, колір, розмір та інтервал вмісту, розбивати вміст на кілька стовпців, додавати анімацію та інші декоративні елементи.
Модуль CSS надає чудову можливість вивчити основи цієї технології, вивчити синтаксис і почати стилізувати свій HTML за допомогою CSS.
CSS є однією з провідних мов у безкоштовній веб-розробці та є стандартизованою специфікацією W3C.
Стандарти CSS діляться на кілька рівнів.
CSS1 зараз вважається застарілим, CSS2.1 рекомендовано для використання, а CSS3, розділений на різні модулі, продовжує розвиватися і стандартизуватися.

4.2 Робота із CSS

Перш за все, нам потрібно повідомити HTML-документу, що ми використовуємо певне правило CSS.
Існує три основних способи застосування CSS до документів HTML.
Розглянемо найпоширеніший і зручний спосіб - використання підключення CSS у заголовку документа.
Це створює файл у тій же папці, що й HTML-документ styles.css.
Розширення .css вказує на те, що це файл CSS.
Щоб зв’язати файл styles.css з index.html, додайте такий рядок десь у <head> в документі HTML (Рис. 4.2.1)

```
<link rel="stylesheet" href="style2.css">
```

Рисунок 4.2.1 – Підключення стилів до документа

4.3 Селектори CSS

Селектори CSS використовуються для форматування елементів HTML на веб-сторінках.

Існує широкий набір селекторів CSS, які дозволяють точно вибрати, які елементи стилізувати.

Селектор — це вираз, який повідомляє браузеру, які HTML-елементи стилізувати за допомогою певних властивостей CSS у блоці стилю.

Знання потрібного вам селектора допоможе вибрати правильний елемент.

Ця група містить селектори елементів HTML, такі як <h1>. (Рис. 4.3.2)

Рисунок 4.3.2 – Селектор HTML-елементів

До групи належать і **селектори класів** (Рис. 4.3.3)

Рисунок 4.3.3 – Селектор класів

Селектори ідентифікаторів **(ID)** (Рис. 4.3.4)

Рисунок 4.3.4 – Селектор класів

Ця група містить псевдокласи, **які** стилізують **певні стани елементів.**

Псевдоклас: Наприклад, :**hover** застосовує **правило лише тоді, коли вказівник миші знаходиться над елементом.**

Приклади псевдокласів:

• : link **–** стиль для посилань, які ще не **переміщено.**

• : visited - стиль для посилань, які вже були **відвідані.**

• : hover **– Стиль** для **плаваючих елементів.**

• : focus – стиль для інтерактивних елементів, які **отримують** фокус **за допомогою клавіші Tab.**

• : active **–** стиль для активованих **елементів.**

• : valid **- стилі** для форм, які успішно пройшли перевірку типу **даних.**

• : invalid **- Стиль** для форм, вміст **яких** не відповідає вказаному **типу.**

• : enabled – **Стилі** для всіх активних **форм.**

• : disabled **-** стиль для **вимкнених** і **неактивних форм.**

• : in-range – стиль для полів, значення яких знаходяться в певному **діапазоні.**

• : out-of-range **- Стиль** для полів, значення яких **виходять за межі зазначеного діапазону.**

• : lang() **–** стиль **елемента, визначеного** для певної **мови.**

• : not(селектор) **- Стиль** для елементів, які не відповідають **указаному селектору.**

• : target **- елементи стилю** з **символами # у посиланнях, які вказують** на інший розділ **сторінки.**

• : checked **-** стиль для **елемента, вибраного користувачем.** (Рис. 4.3.1.1)

```
.box:hover{
  width: 200px;
  color: red;
}
```

Рисунок 4.3.1.1 – Псевдоклас

Селектор псевдокласу структури типу **визначає певний тип дочірнього елемента.**

• : nth-of-type() **- Вибирає елементи за типом.**

• : first-of-type - **Вибирає** перший елемент **зазначеного типу.**

• : last-of-type - вибирає останній елемент **зазначеного типу.**

• : nth-last-of-type() **- Вибирає елементи на основі позиції** та **типу.**

• : **only-of-type – Вибирає** єдиний елемент **указаного** типу серед усіх.

Селектори псевдоелементів використовуються для додавання **спеціальних** символів **або** інформації за допомогою властивості **вмісту.**

• : first-letter **-** вибирає **перший символ** кожного **абзацу та** застосовується **лише** до **елементів блоку.**

• : first-line **- Вибирає** перший рядок кожного **абзацу та** застосовується **лише** до **елементів блоку.**

• : before **– відображає вміст,** який вставляється перед **елементом.**

• : after **– друкує вміст, вставлений** після елемента.

І остання група селекторів дозволяє **комбінувати селектори,** щоб легко знаходити **певні** елементи **в документі.**

У наступному прикладі **використовується дочірній об'єднувач (>), щоб знайти дочірній елемент** <article> (Рис. 4.3.1.2)

```
article > p {
}
```

Рисунок 4.3.1.2 – Об'єднання селекторів

4.4 Блокова модель

Кожен **елемент CSS вкладено в блоки,** і розуміння **того, як ці блоки працюють, є ключовим для оволодіння макетом CSS, який є вкладеністю елементів по відношенню** до інших елементів.

Блокові та **вбудовані** елементи визначаються їхньою поведінкою в контексті потоку сторінки та **по відношенню до** інших елементів на сторінці.

Коли елемент **визначено** як **блок,** він **має** такі **функції:**
• Починається з нового рядка.

• **Розгортається** вздовж **рядка, заповнюючи** весь доступний **простір** у контейнері.
Зазвичай це означає, що блок **має** таку саму ширину, **що й** його контейнер, **і займає** 100% доступного простору.

• Застосовуються властивості width **та** height**.**

• Зовнішні та внутрішні **виїмки дозволяють рамі відводити** інші елементи від **себе.**

Створюючи блокові елементи в CSS, враховуйте такі компоненти: • Вміст: Це область, де **відображатиметься вміст.**

Ви можете налаштувати розмір цієї області за допомогою **таких** властивостей, як width та height**.**

• **Внутрішнє доповнення: Визначає доповнення** навколо вмісту як **пробіл.**

Розмір цих відступів контролюється padding **х.**

• Рамка: Рамка **охоплює** вміст **і** відступи.

Ви можете налаштувати розмір і стиль рамки за допомогою **таких** властивостей, як **border.**

• **Зовнішнє заповнення:** Це зовнішній шар, **який** містить вміст, **заповнення та поля.**

Його розмір контролюється за допомогою **таких** властивостей, як margin**.** (Рис. 4.4.1)

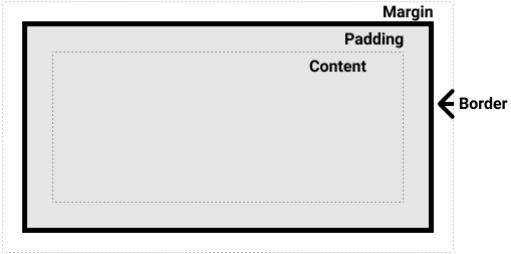


Рисунок 4.4.1 – Компоненти блоку

У типовій блочній моделі зазначення атрибутів Width і Height для елемента встановлює ширину і висоту його вмісту.

Відступи та поля додаються до цих значень, щоб отримати загальний розмір елемента. (Рис. 4.4.2)

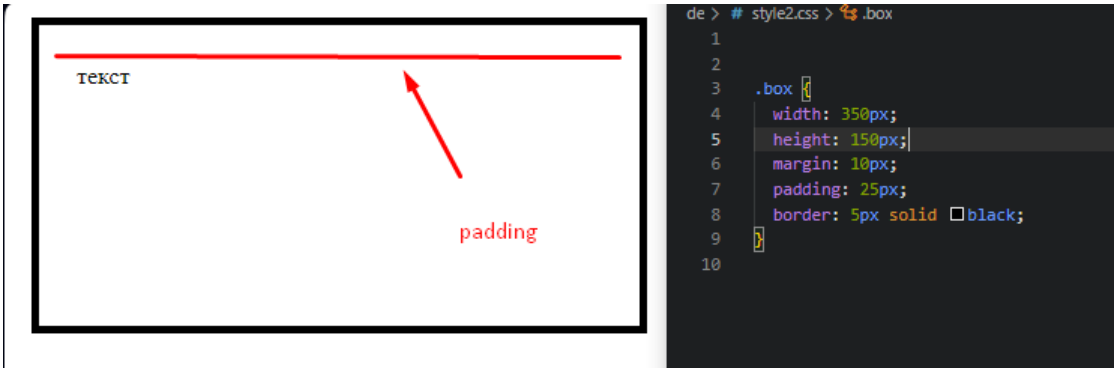


Рисунок 4.4.2 – Внутрішні відступи

Відступ — це невидимий простір, який оточує елемент і віддаляє від нього інші елементи.

Це зміщення може бути позитивним або негативним.

Використання від’ємних значень може спричинити перекриття деяких елементів сторінки.

Зовнішнє доповнення завжди додається після обчислення видимого розміру блоку, незалежно від того, використовується стандартна чи альтернативна модель блоку. (Рис. 4.4.3)

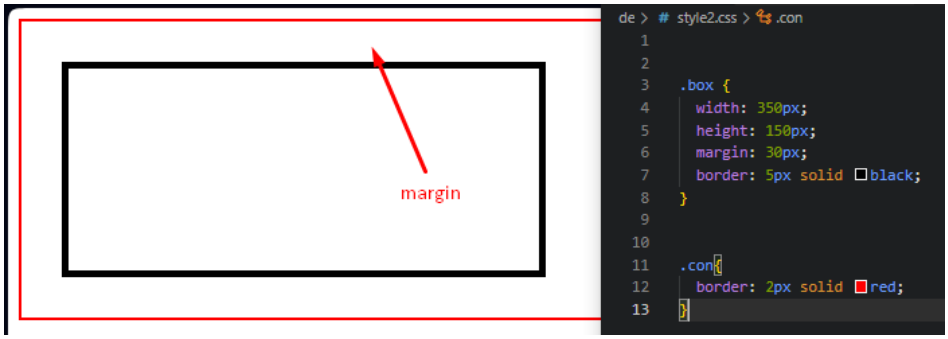


Рисунок 4.4.3 – Зовнішні відступи

4.5 Фон та кордони

Властивість Background Color встановлює колір фону елемента CSS.
Ця властивість приймає будь-який дійсний колір <color>.
Колір, встановлений background-color, застосовується до самого вмісту та відступу (padding). (Рис. 4.5.1) (Рис. 4.5.2)

Рисунок 4.5.1 – Зміна кольору фона

Рисунок 4.5.2 – Фон картинка

Властивість background-repeat **визначає** спосіб повторення фонового зображення.

Його можна **встановити** за допомогою наступних **значень:**

• no-repeat: **запобігає повторенню** фонового зображення **в усіх** напрямках.

• repeat-x: **повторює фонове** зображення **по горизонталі.**

•repeat-y: повторює фонове зображення по вертикалі.

• repeat**: вказує, що фонове** зображення **повторюється** як **горизонтально,** так і **вертикально.**

Це значення встановлено за замовчуванням. (Рис. 4.5.3)

```
background-image: url(/img/fone/Warframe.png);
background-repeat: no-repeat;
```
\\

Рисунок 4.5.3 – Вимкнення повтору фону

4.6 Позиціювання

Розташувавши елементи в документообігу, можна розрізняти різні частини та змінювати їх поведінку.
Покладіть їх один на одного або залиште в одному місці у вікні браузера.
Для елементів HTML доступні різні типи вирівнювання.
Властивість position використовується для визначення певного типу позиціонування для елемента.

Статичне позиціонування — це значення за замовчуванням, надане кожному елементу, яке просто розміщує елемент у стандартному положенні в документообігу. (Рис. 4.6.1)

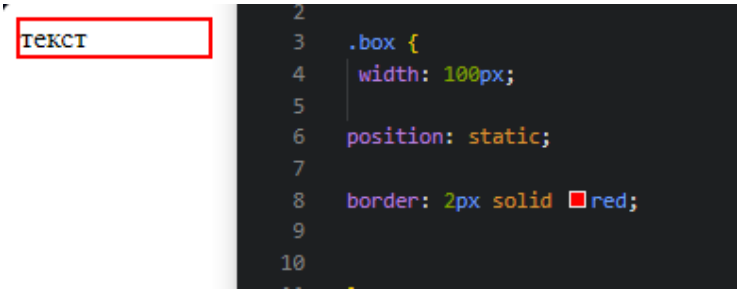


Рисунок 4.6.1 – Позиціювання статичне

Відносне позиціонування - це перший тип позиціонування, який ми розглянемо.
Це дуже схоже на статичне розміщення, за винятком того, що ви можете змінити кінцеве положення розміщених об’єктів і розмістити їх у звичайному макеті.
Це включає в себе можливість накладати інші елементи на сторінку. (Рис. 4.6.1)

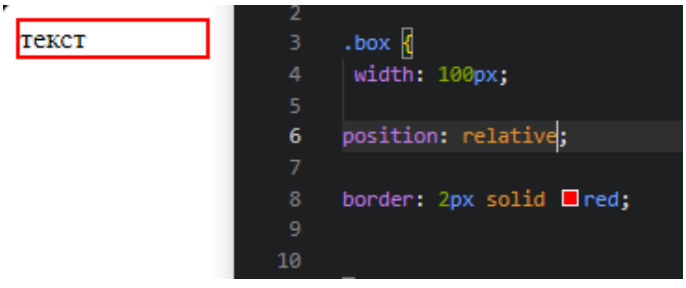


Рисунок 4.6.1 – Позиціювання відносне

Абсолютне позиціонування – отримує елементи з усього потоку, визначаючи їхню нову позицію відносно фіксованого батька.

Якщо такого батьківського елемента немає, нове положення визначається відносно вікна браузера (при цьому ширина елемента встановлюється автоматично відповідно до його вмісту).

Позиція обчислюється на основі властивостей left, top, right та bottom.

Вплив на позицію враховується значенням властивості position батьківського елемента.

Якщо встановлено статично або не має батьківського елемента, координати відраховуються від краю вікна браузера. (Рис. 4.6.2)

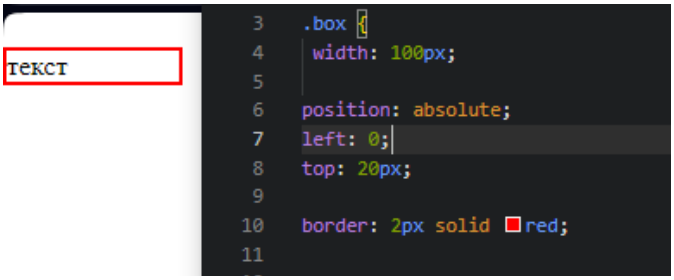


Рисунок 4.6.2 – Позиціювання абсолютне

4.7 Обтікання

Загортання його у властивість float дозволяє переміщувати блок ліворуч або праворуч залежно від бажаного результату.

Плаваючий блок рухається в напрямку, визначеному його властивістю float, доки його кінець не досягне кінця блоку або іншого елемента.

Для правильного використання Float рекомендується вказати ширину плаваючого блоку, щоб залишити простір для додаткового вмісту.

Коли у плаваючого елемента закінчується простір по горизонталі, він починає рухатися вниз, штовхаючи інші елементи навколо себе.

Властивість float визначає основні умови для налаштування поведінки плаваючих блоків.

Розміри блоку враховують поля та відступи, але відступи вмісту не об’єднуються.

4.8 Z-index

Властивість z-index CSS визначає порядок укладання розміщеного елемента, його дочірніх елементів або елементів flex вздовж осі Z.

Елементи з більш високими значеннями z-індексу перекривають елементи з нижчими значеннями z-індексу, щоб визначити порядок, у якому вони відображаються на екрані. (Рис. 4.8.1)

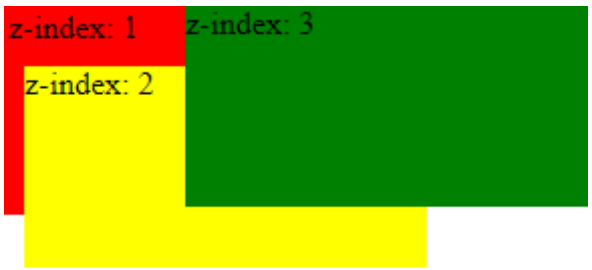


Рисунок 4.8.1 – Порядок накладання

4.9 Робота із текстом

Властивість **Color встановлює** колір тексту **переднього плану** або вмісту **вибраного елемента.**

Ця властивість визначає кольоровий спектр тексту, а також може впливати на інші елементи переднього плану, **наприклад:**

Використовувати текстові прикраси, щоб підкреслити або **накласти текст** за допомогою text-decoration**.** (Рис. 4.9.1)

Рисунок 4.9.1 – Задання кольору тексту

Властивість CSS font-size визначає розмір шрифту певного елемента.
Це дозволяє встановити розмір тексту на веб-сторінці, який зазвичай вказується в одиницях вимірювання, наприклад у пікселях (px). (Рис. 4.9.2)

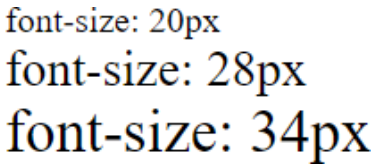


Рисунок 4.9.2 – Задання розміру тексту

Властивість CSS text-decoration визначає стилі оформлення тексту, наприклад:
Підкреслення, лінія над текстом або лінія під текстом.
Ця властивість дозволяє змінювати вигляд тексту в залежності від дизайну сторінки. (Рис. 4.9.3)

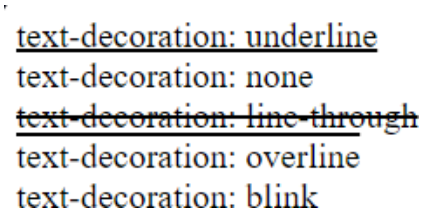


Рисунок 4.9.3 – Властивість text-decoration

Властивість CSS text-align визначає горизонтальне вирівнювання тексту в межах елемента блоку.

Це використовується для визначення способу горизонтального вирівнювання тексту всередині контейнера. (Рис. 4.9.4)

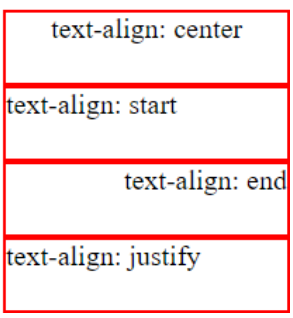


Рисунок 4.9.4 – Властивість text-align

CSS використовує властивість font-weight для визначення щільності шрифту тексту.

Контролює тонкість або товщину тексту, що відображається на веб-сторінках.

Ця властивість приймає число від 100 до 900, такі ключові слова, як «normal» або «bold», або спеціальне значення «bolder». (Рис. 4.9.5)

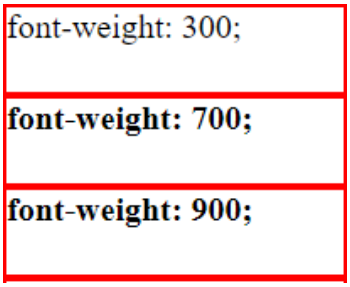


Рисунок 4.9.5 – Властивість font-weight

Властивість CSS line-height визначає відстань між рядками тексту в межах елемента блоку.

Ця властивість дозволяє керувати вертикальним інтервалом між рядками тексту.

Це впливає на зовнішній вигляд і читабельність текстового вмісту в елементі . (Рис. 4.9.6)

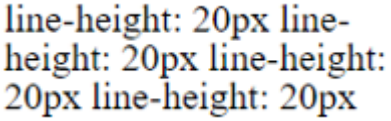


Рисунок 4.9.6 – Властивість line-height

Властивість CSS Letter-spacing встановлює відстань між символами тексту в елементі блоку.

Надає можливість контролювати відстань між символами для досягнення певних дизайнерських ефектів. (Рис. 4.9.7)

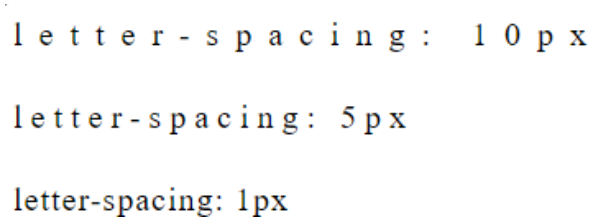


Рисунок 4.9.7 – Властивість letter-spacing

Ви можете використовувати властивість CSS text-shadow, щоб додати тіні до тексту в елементах блоку.

Ця властивість дозволяє створювати різні тіньові ефекти, які роблять текст більш видимим і привабливішим на веб-сторінках. (Рис. 4.9.8)

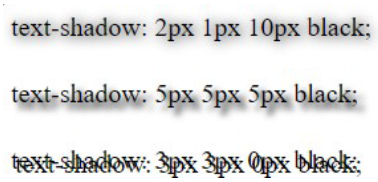


Рисунок 4.9.8 – Властивість text-shadow

Властивості перетворення тексту CSS дозволяють змінювати відображення тексту з урахуванням регістру.

Це допоможе вам стилізувати текстовий вміст відповідно до ваших вимог дизайну.

Синтаксис буде таким (Рис. 4.9.9)

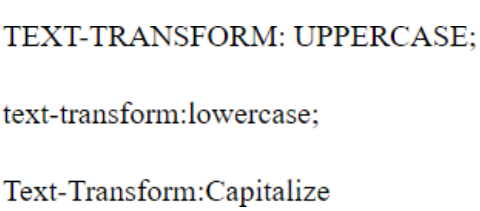


Рисунок 4.9.9 – Властивість text-transform

5 JAVASCRIPT ЯК РЕСУРС ДЛЯ ПОКРАЩЕННЯ САЙТУ

5.1 Основні поняття JavaScript

JavaScript — це мова програмування, яка дозволяє реалізовувати складні взаємодії на веб-сторінках.

Щоразу, коли ви переглядаєте веб-сторінку, ви бачите більше, ніж просто статичний вміст.

За допомогою JavaScript ви можете додавати динаміку, оновлювати вміст у реальному часі, вбудовувати інтерактивні карти та створювати 2D/3D-анімацію.

Ядро JavaScript містить набір стандартних функцій, які дозволяють виконувати наступне: • Маніпулювання DOM (об’єктною моделлю документа): JavaScript дозволяє динамічно змінювати структуру та вміст HTML-документів для створення веб-сторінок із живою інтерактивністю.

.

• Обробка подій: мови дозволяють створювати відповіді на такі події, як клацання, натискання клавіш і завантаження сторінки, що дозволяє створювати інтерактивні інтерфейси.

• Змінні: Такі ключові слова, як var, let і const, дозволяють створювати змінні в JavaScript і використовувати їх для зберігання даних.

• Структури даних: Вбудовані структури даних, такі як масиви та об’єкти, допомагають ефективно організувати та обробляти інформацію.

• Функції: JavaScript підтримує визначення та виклик функцій, дозволяючи вам структурувати та повторно використовувати свій код.

• Асинхронне програмування: Використовуючи обіцянки та функції зворотного виклику, JavaScript ефективно обробляє асинхронні операції, такі як: Отримати дані з сервера.

• Обробка помилок: JavaScript має ефективний механізм обробки винятків для обробки ситуацій помилок.

• Взаємодія із запитами HTTP: Об’єкт XMLHttpRequest або метод Fetch API дозволяє JavaScript взаємодіяти з сервером, надсилаючи й отримуючи дані асинхронно.

• Взаємодія з API браузера: JavaScript може використовувати різні API браузера для доступу до різноманітних функцій, зокрема:

Робота з локальним сховищем, геолокацією, камерами тощо.

5.1.1 Відомості про DOM

DOM (Document Object Model) — це структура, яка представляє структуру документа HTML або XML у формі дерева об’єктів, до якого можна отримати доступ для взаємодії з JavaScript або іншими мовами програмування.
DOM надає програмістам зручний інтерфейс для динамічної зміни вмісту, структури та стилю веб-сторінок.
Основні концепції та компоненти DOM: • Документ: Представляє весь документ і його вміст.
Це головний об'єкт, який є відправною точкою для всієї структури DOM.
• Елемент: представляє тег HTML на сторінці такі як , <p>.
• Атрибути: Атрибути елемента, такі як атрибути "class" та "id".
Ці характеристики можна змінювати та читати за допомогою JavaScript.
• Текстовий вузол: представляє текстовий вміст в елементі .
• Вузол атрибута: представляє атрибут елемента .
• Методи: методи дозволяють вам взаємодіяти з DOM.
Приклад: getElementById, getElementsByTagName.

Операції, які можна виконувати над DOM:

• Змінити вміст: Додавати, видаляти або змінювати елементи та їхній вміст.
• Зміна атрибутів: Можливість змінювати значення атрибутів елемента відкриває можливість динамічно змінювати властивості елемента.
• Зміни стилю: Ви можете налаштувати вигляд своїх веб-сторінок у реальному часі, динамічно змінюючи стилі своїх елементів.
• Взаємодія з подіями: Додайте обробники подій, наприклад клацання та наведення курсора.
• Змінити структуру: змініть структуру дерева DOM, додавши або видаливши елементи.

5.1.2 Обробник подій

JavaScript надає можливість призначати обробники подій, які визначають поведінку вашого коду у відповідь на різні дії користувача або системи.
Основні концепції обробки подій: • Визначення обробника подій: Обробник події — це функція, яка викликається у відповідь на певну подію.

Наприклад, клацання мишею, натискання клавіш, завантаження сторінки тощо.
• Об'єкт події: Кожен обробник події отримує об'єкт події, який містить інформацію про саму подію.
• Видалення обробника: Щоб уникнути потенційних конфліктів і звільнити ресурси, ви можете видалити обробник події за допомогою методу removeEventListener.
• Припинення обробки подій: Щоб зупинити типову поведінку браузера, скористайтеся методом event.preventDefault().

5.1.3 Змінні

Змінні в JavaScript діють як контейнери для значень, таких як числа, які можна використовувати в математичних операціях, або рядки, які можна використовувати як текстові дані в складних структурах.
Однак характерною рисою змінних є те, що їх значення можуть змінюватися під час виконання програми. (Рис. 5.1.3.1)

```
<button>Text</button>
```

```
78
79  const button = document.querySelector("button");
80
81
```

Рисунок 5.1.3.1 – Оголошення змінної JS, яка містить зв’язок із button, яка оголошена в HTML

5.1.4 Структура даних

JavaScript має кілька структур даних.
Іншими словами: Масив у JavaScript — це впорядкована колекція значень, яка може містити елементи будь-якого типу даних. (Рис. 5.1.4.1)

```
let myArray = [1, 'Hello', true, 3];
```

Рисунок 5.1.4.1 – Оголошення масиву

Об’єкт у JavaScript — це невпорядкована колекція пар ключ-значення.
Ключі можуть бути рядками або символами, а значення можуть бути будь-якими типами даних. (Рис. 5.1.4.2)

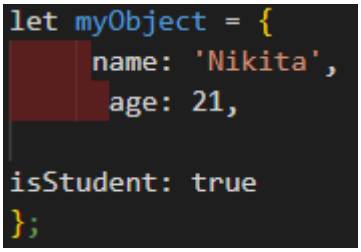


Рисунок 5.1.4.2 – Оголошення об’єкта

Стрічка в JavaScript – це послідовність символів, яку обмежують лапки. (Рис. 5.1.4.3)

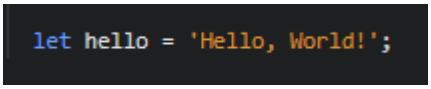


Рисунок 5.1.4.3 – Оголошення стрічки

Числа включають цілі числа та числа з плаваючою комою. (Рис.5.1.4.4)

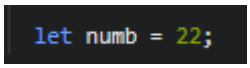


Рисунок 5.1.4.4 – Оголошення числа

Мапа в JavaScript - це структура даних, яка представляє собою колекцію пар ключ-значення, де ключі можуть бути представлені будь-яким типом даних. (Рис. 5.1.4.5)

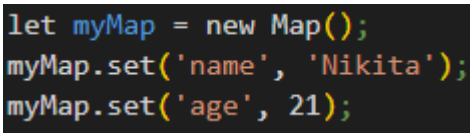


Рисунок 5.1.4.5 – Оголошення мапи

5.1.5 Функції

Функція на мові JavaScript — це особливий тип об’єкта, який дозволяє формалізувати поведінку та логіку обробки даних у межах мови програмування. Оголошення функції містить ключове слово функції та такі елементи: • Назва функції.
• Список параметрів, які приймає функція, укладено в дужки ().
• Інструкції, які виконуються після виклику функції, укладаються у фігурні дужки { }.(Рис. 5.1.5.1)

```
function square(number) {
  return number + number;
}
```

Рисунок 5.1.5.1 – Оголошення функції

Існують різні види функцій JavaScript:

- Оголошення функції (Function Declaration) (Рис. 5.1.5.2)

```
function myFunction(parameter1, parameter2) {
  // тіло функції
}
```

Рисунок 5.1.5.2 – Оголошення функції Declaration

- Функціональний вираз (Function Expression) (Рис. 5.1.5.3)

```
const myFunction = function(parameter1, parameter2) {
    // тіло функції
};
```

Рисунок 5.1.5.3 – Оголошення функції Expression

- Анонімна функція (Anonymous Function) (Рис. 5.1.5.4)

```
const myFunction = function(parameter1, parameter2) {
    // тіло функції
};
```

Рисунок 5.1.5.4 – Оголошення функції Anonymous

Функція зворотного виклику (Callback Function) (Рис. 5.1.5.5)

```
function SUMAXY(x, y, suma) {
    const result = x + y;
    suma(result);
}

SUMAXY(2, 3, function(result) {
    console.log('Результат операції:', result);
});
```

Рисунок 5.1.5.5 – Оголошення функції Callback

- Стрілкова функція (Arrow Function) (Рис. 5.1.5.6)

```
const addNumbers = (a, b) => a + b;
```

Рисунок 5.1.5.6 – Оголошення функції Arrow

5.1.6 Підключення JavaScript

JavaScript використовується **на сторінках HTML подібно до** CSS.

Подібно до того, як CSS використовує елементи <link> для зовнішніх стилів і <style> для **вбудованого** HTML, JavaScript **вимагає** лише **одного елемента** у світі **HTML:** елемент <script>.

Є два **способи підключити** JS до **документів** HTML.

Перший варіант **– вставити код безпосередньо** в **HTML-документ** (Рис. 5.1.6.1)

Рисунок 5.1.6.1 – Підключення напряму

Другий варіант, це підключення зовнішнього файлу, в якому і буде писатисявесь код (Рис. 5.1.6.2)

Рисунок 5.1.6.2 – Підключення зовнішнього файлу

5.1.7 JQuery для полегшення роботи з JavaScript

jQuery — це бібліотека, розроблена на основі мови програмування JavaScript.
jQuery був представлений у 2006 році і зараз широко використовується на різноманітних вебсайтах, особливо на платформі WordPress.
Ви можете розглядати jQuery як структуру, яка надає багато інструментів для ефективної роботи з веб-документами.
Ця бібліотека дозволяє легко вирішувати різноманітні типові завдання, зокрема: Наприклад, перевірка форм, створення слайдерів тощо.
jQuery дозволяє вирішувати такі завдання набагато швидше, ніж чистий JavaScript, завдяки різноманітним готовим інструментам.
Важливі особливості jQuery: • Використання різних правил таблиць стилів CSS.
• Обробка подій у документах.
• Створюйте анімацію та застосовуйте різні ефекти до ваших документів.
• Маніпулювання та відображення об’єктів у структурі DOM сторінки.
• Застосування технології AJAX.

5.1.8 Селектори jQuery

Селектори jQuery використовуються для вибору та керування елементами HTML.
Розглянемо базовий селектор.
Селектор елементів jQuery вибирає елементи на основі імені елемента.
У цьому прикладі натисніть.
усі елементи «p» будуть приховані. (Рис. 5.1.8.1)

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

Рисунок 5.1.8.1 – Приклад селектора елементів

Селектор #id jQuery використовує атрибут id тегу HTML для пошуку певного елемента.

У цьому прикладі елемент із #id зникає після натискання кнопки. (Рис. 5.1.8.2)

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

Рисунок 5.1.8.2 – Приклад селектора #id

Селектор .class jQuery використовується для пошуку елементів певного класу.

У цьому прикладі елементи, призначені цьому класу, зникають після натискання кнопки. (Рис. 5.1.8.3)

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
```

Рисунок 5.1.8.3 – Приклад селектора .class

5.1.9 Події

Ви можете використовувати події в бібліотеці jQuery, щоб реагувати на дії користувача або зміни документа.

Робота з подіями передбачає налаштування обробників подій, які викликаються, коли відбувається певна подія.

Покращуйте свою взаємодію з HTML за допомогою JQuery.

Якщо ви хочете використовувати обробник подій, ви повинні встановити обробник подій за допомогою методу .on() або скороченої форми, такої як .click() або .hover().(Рис. 5.1.9.1)

```
$('#myElement').on('click', function() {
    // Код, який викликається при кліку на елемент
});
```

Рисунок 5.1.9.1 – Приклад встановлення обробника для події кліку

В кожному обробнику подій передається об'єкт події, що містить деталі щодосамої події. (Рис. 5.1.9.2)

```
$('#myElement').on('click', function(event) {
    console.log('Координати кліку:', event.pageX, event.pageY);
});
```

Рисунок 5.1.9.2 – Приклад використання об'єкта події для отримання координат кліку.

Метод event.preventDefault() використовується для блокування типової поведінки браузера, пов’язаної з певною подією.

Ось приклад того, як це використовувати для зупинки посилання (Рис. 5.1.9.3)

```
$('#myLink').on('click', function(event) {
    event.preventDefault();
    // Код, який викликається при кліку, зупиняючи перехід за посиланням
});
```

Рисунок 5.1.9.3 – Приклад використання для зупинки переходу за посиланням.

5.1.10 Під’єднання JQuery

Щоб включити бібліотеку jQuery **на свій вебсайт, додайте** посилання **на** <head> **свого** HTML-документа.

Ось приклад тегу <script>, який **може підключитися до** jQuery з офіційного CDN. (Рис. 5.1.10.1)

```
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" href="style2.css">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
</head>
```

Рисунок 5.1.10.1 – Приклад підключення JQuery

Також, можна підключити локальний файл, якщо завантажити бібліотеку на свій ПК. (Рис. 5.1.10.2)

```
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" href="style2.css">
  <script src="шлях_до_файлу/jquery.js"></script>
</head>
```

Рисунок 5.1.10.2 – Приклад підключення JQuery локально

6 Роль PHP у створенні вебсайту

PHP — це мова програмування загального призначення, яка зазвичай використовується для створення веб-додатків і сценаріїв.

Використовується для розробки динамічних вебсайтів, обробки веб-форм, взаємодії з базами даних і вирішення різноманітних завдань у сфері веб-розробки.

Основними можливостями PHP є: • Створення динамічних вебсайтів: PHP дозволяє розробляти веб-сторінки, які динамічно змінюються у відповідь на дії користувача та інші фактори, що полегшує створення інтерактивних і динамічних вебсайтів.

• Обробка форм: Зручна інтеграція з формами HTML дозволяє PHP легко обробляти дані, введені користувачами через веб-форми.

Це ключ до реалізації функцій зворотного зв'язку, схвалень і збереження інформації про користувачів.

• Взаємодія з базою даних: PHP надає інструменти для підключення до різних систем керування базами даних і виконання таких операцій з даними, як додавання, читання, оновлення та видалення.

• Обробка зображень: Вбудована обробка зображень і можливості маніпулювання роблять PHP ефективним інструментом для розробки графічних програм і маніпулювання мультимедійним вмістом.

• Файлові операції: PHP дозволяє читати та записувати файли на вашому сервері, спрощуючи керування файловою системою та зберігання даних.

• Створення файлів cookie та сеансів: Ця мова може використовувати файли cookie та сеанси для збереження стану інформації в різних запитах користувачів.

• Обробка помилок: PHP забезпечує ефективні засоби обробки помилок, щоб ви могли видавати корисні повідомлення та застосовувати заходи безпеки.

• Взаємодія з XML: PHP може взаємодіяти з документами XML, обробляти та створювати їх.

6.1 Змінні в PHP

Змінні використовуються для зберігання та обробки даних у PHP.

Основними характеристиками змінних у PHP є: Іменування: Імена змінних PHP починаються зі знака долара ($) і можуть містити літери, цифри та підкреслення.

Має бути чутливим до регістру (наприклад, різниця між $var і $Var). (Рис. 6.1.1)

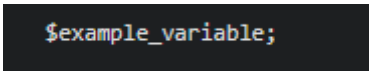


Рисунок 6.1.1 – Іменування

Присвоєння значень: Значення призначаються змінним за допомогою оператора присвоєння (=). (Рис. 6.1.2)

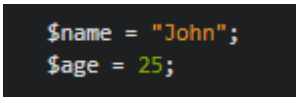


Рисунок 6.1.2 – Присвоєння значення

Типи даних: PHP визначає слабо типізовану мову.

Це означає, що тип даних змінної може змінюватися під час виконання програми.

У цій мові типи даних можуть бути примітивними.

Приклади: рядки (string), цілі числа (integer), числа з плаваючою комою (float), логічні значення (boolean), або комплексні числа (array).

Приклади: масиви, об'єкти тощо. (Рис. 6.1.3)

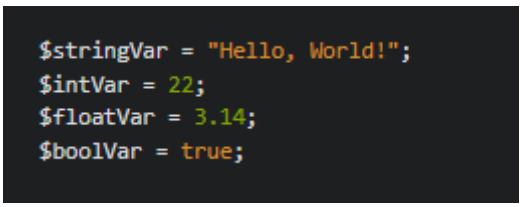


Рисунок 6.1.3 – Різні типи даних

Динамічні змінні: **PHP дозволяє динамічно** використовувати **змінні** для доступу до **імен функцій і** властивостей **об'єктів.**

Глобальні та локальні змінні: **у** PHP **глобальні та локальні змінні** можуть бути оголошені на рівні глобального контексту (поза функціями та класами) або на рівні локального контексту **(всередині** функцій **і** блоків коду). (Рис. 6.1.4)

```php
$globalVar = "глобальна змінна";

function exampleFunction() {
    $localVar = "локальна змінна";
    global $globalVar;
}
```

Рисунок 6.1.4 – Оголошення глобальних і локальних змінних

6.2Функції PHP

У PHP функція — це блок коду, який виконує певне завдання і може бути викликаний з інших частин програми.

Найважливіші функції PHP:

Оголошення функції (Рис. 6.2.1)

```php
function myFunction($param1, $param2) {

}
```

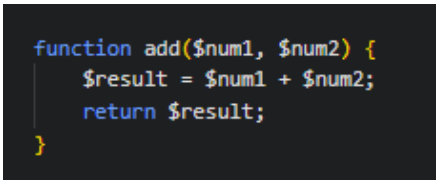
Рисунок 6.2.1 – Оголошення функції

Параметри функції (Рис. 6.2.2)

```php
function add($num1, $num2) {
    return $num1 + $num2;
}
```

Рисунок 6.2.2 – Параметри функції

Повернення значень за допомогою return (Рис. 6.2.3)



```php
function add($num1, $num2) {
    $result = $num1 + $num2;
    return $result;
}
```

Рисунок 6.2.3 – Повернення значень

7 Етапи створення вебсторінки

7.1 Робота з редактором Visual Studio Code

Visual Studio Code (VS Code) — це безкоштовний текстовий **редактор із відкритим кодом,** розроблений Microsoft.

Він має **широкий набір функцій,** включаючи вбудовану підтримку **різних** мов програмування, розширень **та** інструментів, **які** полегшують **написання,** редагування **та налагодження коду.**

З допомогою Visual Studio Code розробимо код для знаходження предметів. (Рис. 7.1.1)

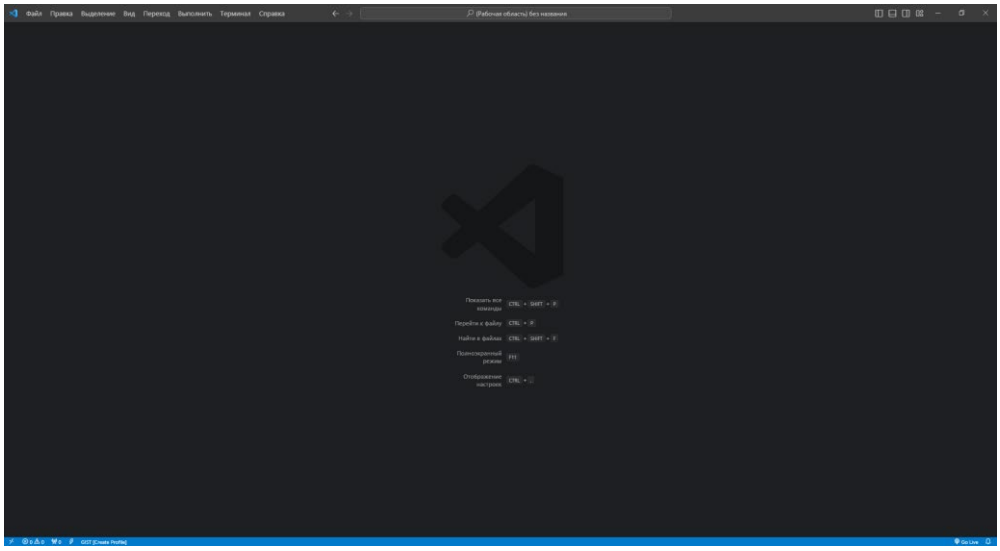


Рисунок 7.1.1 – Інтерфейс програми Visual Studio Code

Перший етап, це створення трьох файлів це:

- Index.html – який буде відповідати за структуру нашого елемента.
- Server.js – допоможе оформити нашу сторінку.
- Package та Package-lock – завантаження залежності.

7.2.1 Створення структури документа

Розпочнемо роботу із файлом Index.html, розробимо сторінку заголовка,

вказуємо кодування символів сторінки UTF-8, задаємо мову сторінки, зробимо відображення веб-сторінки на різних пристроях (Рис. 7.2.1.1)

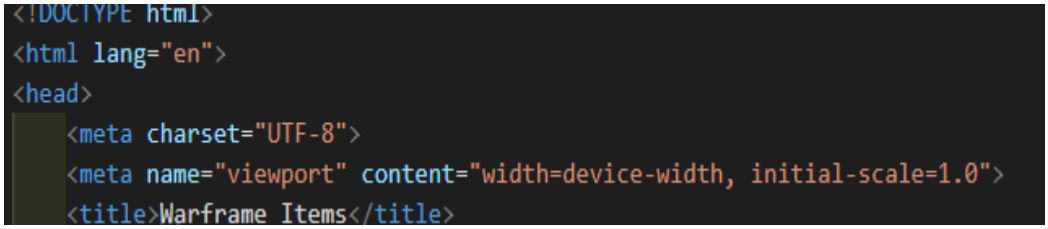


Рисунок 7.2.1.1 Інтерфейс програми Visual Studio Code

Переходимо до розробки стилів CSS, який описує вигляд веб-інтерфейса на сторінці, що використовує HTML. (Рис. 7.2.1.2)

```
    body {
        font-family: Arial, sans-serif;
        margin: 0;
        padding: 20px;
        background-color: #f4f4f4;
    }
    h1 {
        text-align: center;
    }
    #categories {
        margin: 20px 0;
        text-align: center;
    }
    #items {
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
    }
    .item {
        background-color: white;
        border: 1px solid #ddd;
        border-radius: 5px;
        margin: 10px;
        padding: 10px;
        width: 200px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        text-align: center;
    }
    .item img {
        max-width: 100%;
        height: auto;
        border-radius: 5px;
    }
    .item h3 {
        margin: 10px 0;
    }
</style>
```

Рисунок 7.2.1.2 Стиль для адаптивної платформи

Розробимо кнопку отримання та відображення елементів для полегшення знаходження предметів у грі Warframe. Блоки ми будемо створювати за допомогою елементу div. В нашій структурі це буде вибір предметів. (Рис. 7.2.1.3)

```
<h1>Warframe Items</h1>
<div id="categories">
    <select id="categorySelect">
        <option value="warframes">Warframes</option>
        <option value="weapons">Weapons</option>
        <option value="mods">Mods</option>
        <option value="resources">Resources</option>
        <option value="archwing">Archwing</option>
        <option value="pets">Pets</option>
        <option value="relics">Relics</option>
        <option value="fish">Fish</option>
    </select>
    <button onclick="fetchItems()">Get Items</button>
</div>
<div id="items"></div>
```

Рисунок 7.2.1.3 кнопку отримання та відображення елементів

Розробимо функціональність завантаження і відображення елементів із сервера залежно від вибраної категорії на JavaScript. Цей скрип містить дві основні функції: fetchItems та displayItems. (Рис. 7.2.1.4)

```
<script>
    async function fetchItems() {
        const category = document.getElementById('categorySelect').value;
        const response = await fetch(`http://localhost:3000/api/${category}`);
        const items = await response.json();
        displayItems(items);
    }

    function displayItems(items) {
        const itemsContainer = document.getElementById('items');
        itemsContainer.innerHTML = '';

        items.forEach(item => {
            const itemDiv = document.createElement('div');
            itemDiv.className = 'item';

            const imageUrl = `https://cdn.warframestat.us/img/${item.imageName}`;

            itemDiv.innerHTML = `
                <img src="${imageUrl}" alt="${item.name}">
                <h3>${item.name}</h3>
                <p>${item.description || 'No description available'}</p>
            `;

            itemsContainer.appendChild(itemDiv);
        });
    }
</script>
```

Рисунок 7.2.1.4 функціональність завантаження і відображення елементів

7.3.1 Обробка та надання відповіді від API

Створимо та підключемо необхідних модулів. Це вбудованні модулі Node.js, такі як http та fs (для роботи з файловою системою), а також зовнішні модулі, які встановлені за допомогою npm, такі як express (фреймворк для веб-додатків).Ми використовуемо Node.js для встановлення Express.js через npm та за допомогою косолі робимо папку warframe-api (Рис. 7.3.1.1)

```
mkdir warframe-api
cd warframe-api
```

```
npm install express
```

Рисунок 7.3.1.1 встановлення Express.js

Після встановлення Express.js створюємо новий каталог для проекту за допомогою теміналу (Рис. 7.3.1.2)

```
npm init -y
```

Рисунок 7.3.1.2 створення нового каталога

Далі створюємо файл server.js. В ньмоу імпортуємо та створюємо express (Рис. 7.3.1.3)

```
const express = require('express');
const Items = require('warframe-items');

const app = express();
```

Рисунок 7.3.1.3 створення express у файлі server.js

Додаемо маршрути для обробки запитів GET та помилок до різних категорій (Рис. 7.3.1.4)

```
app.get('/api/:category', (req, res) => {
    const category = req.params.category;
    const categoryItems = allItems.filter(item => item.category === categories[category]);
    if (categoryItems) {
        res.json(categoryItems);
    } else {
        res.status(404).json({ error: 'Category not found' });
    }
});
```

Рисунок 7.3.1.4 обробка запитів GET та помилок до різних категорій

Встановити порт для з'єднання сервера з сайтом (Рис. 7.3.1.5)

```
app.listen(port, () => {
    console.log(`Warframe API listening at http://localhost:${port}`);
});
```

Рисунок 7.3.1.5 з'єднання порта

Далі ми запускаемо сервер командою node server.js для тестування API (Рис. 7.3.1.6)

```
http://localhost:3000/api/items/weapons
```

Рисунок 7.3.1.6 запус сервера

Тепер, можна спостерігати результат створенної сторінки, за допомогою мвои html та css на (Рис. 7.3.1.7-7.3.1.8)

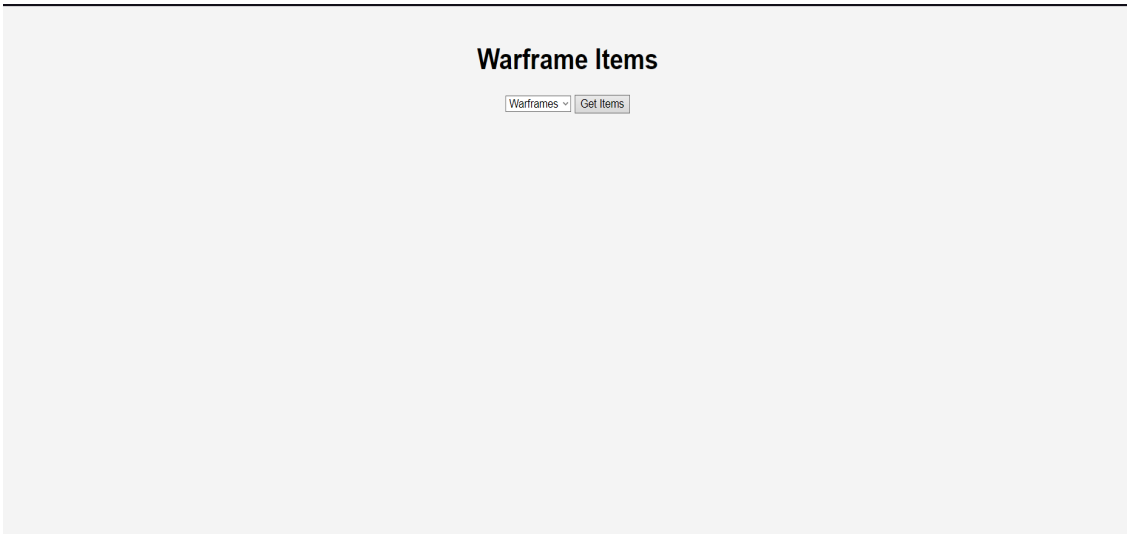


Рисунок 7.3.1.7 Головна сторінка

```
[
  {
    "abilities": [
      {
        "uniqueName": "/Lotus/Powersuits/PowersuitAbilities/GlaiveAbility",
        "name": "Shuriken",
        "description": "Launches a spinning blade of pain, dealing high damage and impaling enemies to walls.",
        "imageName": "shuriken.png"
      },
      {
        "uniqueName": "/Lotus/Powersuits/PowersuitAbilities/SmokeScreenAbility",
        "name": "Smoke Screen",
        "description": "Drops a smoke bomb that stuns enemies and obscures their vision, rendering Ash invisible for a
short time.",
        "imageName": "smoke-screen.png"
      },
      {
        "uniqueName": "/Lotus/Powersuits/PowersuitAbilities/TeleportToAbility",
        "name": "Teleport",
        "description": "Ash teleports towards the target, bringing him into melee range and making enemies vulnerable
to finishers.",
        "imageName": "teleport.png"
      },
      {
        "uniqueName": "/Lotus/Powersuits/PowersuitAbilities/NinjaStormAbility",
        "name": "Blade Storm",
        "description": "Project fierce shadow clones of Ash upon groups of distant enemies. Join the fray using
Teleport.",
        "imageName": "blade-storm.png"
      }
    ],
    "armor": 105,
    "aura": "madurai",
    "bpCost": 35000,
    "buildPrice": 25000,
    "buildQuantity": 1,
```


Рисунок 7.3.1.8 Сторінка з запитом предметів та описом

ВИСНОВОК

Процес розробки мов програмування для веб-розробки відкриває багато можливостей для створення різних вебсайтів.

До них належать торговельні вебсайти, особисті портфоліо, інформаційні магазини, інформаційні платформи або розважальні ресурси.

Однією з основних мов веб-розробки є HTML, який створює базову структуру документів.

Це дозволяє розділити вміст на блоки та ефективно розташувати елементи на сторінці.

CSS відіграє важливу роль у дизайні контенту, роблячи сторінки візуально привабливими та простими у використанні.

Створення API для Warframe - це цікавий і корисний досвід, який дозволяє краще зрозуміти принципи роботи веб-серверів та розробки програмного забезпечення взагалі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Владілен Мінін. JavaScript з Нуля - Курс для початківців с практикой БЕЗ ВОДИ, 2023. *YouTube*. URL: https://www.youtube.com/watch?v=fcMcf_4PjfI
2. Основи застосування UML. Хто та як його використовує. Школа системного аналізу та проектування.URL: https://systems.education/who-uses-uml
3. cMQCq1. Що таке API? Просте пояснення від Петра Газарова. *dev.ua*. URL: https://dev.ua/news/chto-takoe-api-prostym-yazykom
4. Contributors to Вікі Warframe. WARFRAME. *Вікі Warframe*. URL: https://warframe.fandom.com/uk/wiki/WARFRAME
5. GodzzilaS. API. *Warframe Forums*. URL: https://forums.warframe.com/topic/1303173-api/

ПРЕЗЕНТАЦІЯ

Державний університет
інформаційно-комунікаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Розробка адаптивної платформи на JavaScript з алгоритмами AI для аналізу даних онлайн-ігор та рекомендацій

Виконав студент 4 курсу
групи ІСД-42
Биков Микита Роланович
Керівник роботи: викладач
Шахматов Іван Олександрович

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи:** Створення адаптивної платформи на язику програмування JavaScript з алгоритмами штучного інтелекту для спрощення знаходження предметів у грі Warframe
- **Об'єкт дослідження:** онлай-гра «Warframe»
- **Предмет дослідження:** соціальна взаємодія у грі, розвиток навичок, економічні аспекти, технічні аспекти та інше.

Warframe

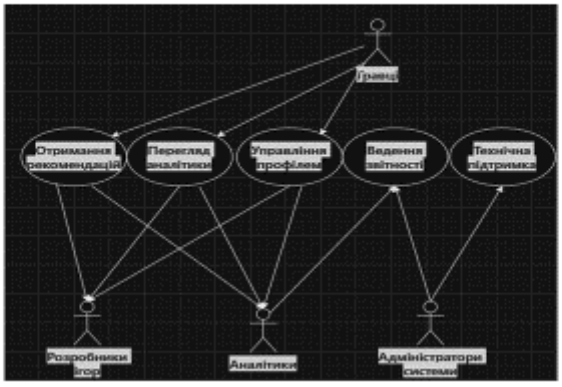
- Відеогра у жанрі науково-фантастичного шутера від третьої особи, розроблена канадською студією Digital Extremes.
- Гра була випущена у 2013 році.
- Гравці керують воїнами Тенно, які використовують Варфрейми — спеціалізовані бойові костюми з унікальними здібностями.
- Гравці можуть виконувати місії в різних світах і регіонах сонячної системи, як самостійно, так і спільно з іншими гравцями.
- Гра має сильний акцент на кастомізацію, дозволяючи гравцям налаштовувати свої Варфрейми, зброю та інший ігровий інвентар через систему модів.
- Гра фінансується на моделі "free-to-play" з використанням мікротранзакцій, де гравці можуть купувати внутрішньоігрову валюту, звану платиною, для придбання різних вдосконалень, швидшого прогресу або косметичних предметів.

ТЕХНІЧНІ ЗАВДАННЯ

- **Вибір технологій**: Вибір фронтенд та бекенд технологій, таких як React для фронтенду та Node.js для бекенду, з урахуванням їхньої придатності для реалізації алгоритмів AI.
- **Архітектура системи**: Проектування архітектури системи з використанням розподіленої архітектури(Сервери для обробки даних AI, сервери для зберігання даних та сервери для керуванням рекомендаціями) для забезпечення масштабованості та надійності.
- **Реалізація функціоналу**: Розробка основних функцій, таких як аналіз даних та знаходження предметів у грі на фронтенді.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

ЗАГАЛЬНА ДІАГРАМА ВИКОРИСТАННЯ(draw.io)



ІНТЕРФЕЙС РОЗРОБЛНОЇ СТОРІНКИ

ID	Thumbnail	Item Name	URL Name
54aae292e7758f0080641575	items/images/en/thumbs/secura_dual_cestra.3e07a4ec6875ff714bb0da8b18c63e0e.128x128.png	Secura Dual Cestra	secura_dual_cestra
54ca90abe7756815c1c11e10	items/images/en/thumbs/creeping_bullseye.d0d8c92 fc08c8be2a2632f37c79b16532f.128x128.png	Creeping Bullseye	creeping_bullseye
54d4c727e7756829 1cc7d753	items/images/en/thumbs/mutalist_alad_v_assassinate_key.7540fb6d7ef2b2843352e52c9f84e73.128x128.png	Mutalist Alad V Assassinate (Key)	mutalist_alad_v_assassinate_key
54e0c9eee7756900744178b0	items/images/en/thumbs/irradiating_disarm.cffd5ed77e1365bdcfd3e8559bd66ecc6.128x128.png	Irradiating Disarm	irradiating_disarm
54e844fbf7756975845a66cf	items/images/en/thumbs/antimatter_absorb.1432c8571d56b6f5aec2504d6abec464f4.128x128.png	Antimatter Absorb	antimatter_absorb
551b65aeee7756f9729e141fc0	items/images/en/thumbs/arcane_barrier.c4dc8c25f0cc8c6f74e4bf0f2f6986c6f5bb.128x128.png	Arcane Barrier	arcane_barrier
551b65c8ef7756ff72b41a2f2206	items/images/en/thumbs/arcane_ice.d3c0d52ba9f923cb3198b3545fd57e57f16.128x128.png	Arcane Ice	arcane_ice
55ac7bf12ef7756ff2b49b0d8ff0	items/images/en/thumbs/telos_boltor.9a108e6b68807c1d61b2402ef703124365.128x128.png	Telos Boltor	telos_boltor
567f30f2b4b8df0432dc8f99c	items/images/en/thumbs/frost_prime_set.49ff5602f0ae1a6aa8fa6e6ec0e87cb21.128x128.png	Frost Prime Set	frost_prime_set
56a7b2a01133866bcb065b93	items/images/en/thumbs/catalyzer_link.3ffa620e87c62e100d51ccc1ceed8d7f65.128x128.png	Catalyzer Link	catalyzer_link
56a7b2bb1133866bcb065c98	items/images/en/thumbs/targeting_subsystem.bab0126cb3161156d1c4c66686e75af8e8.128x128.png	Targeting Subsystem	targeting_subsystem
56aba2f70df6cf83da42400c0	sub_icons/weapon/generic_receiver_128x128.png	Sirun Wealth Receiver	sirun_wealth_receiver
56b656d1ef6086c7e4066081	items/images/en/thumbs/tectonic_fracture.7c38025ee84d0e8617562f5300dd0a712.128x128.png	Tectonic Fracture	tectonic_fracture
57222ffc62ccc365777a67c0f	items/images/en/thumbs/mantis_set.c4718b0defc2695308e68c0a41b9d655.128x128.png	Mantis Set	mantis_set
5729bd02cc059ed5a98c484e6c	items/images/en/thumbs/acid_shells.b43d8982ae6587941c0067032d52314.128x128.png	Acid Shells	acid_shells
5729bd3acb89ed5a98c484e8e	items/images/en/thumbs/nightwatch_napalm.b05aadcfc7dbdb7f0a8e8adb5da7ac584e.128x128.png	Nightwatch Napalm	nightwatch_napalm
57639f08acf86d7037f0c89e62	items/images/en/thumbs/quasar_drill.b72b823f1330e11a83cbe6cb92f5acbe.128x128.png	Quasar Drill	quasar_drill
57639f0facf86d7037f0c89e63	items/images/en/thumbs/comet_blast.8878554f3f52fdb9df2086e4ecf3ac7a9.128x128.png	Comet Blast	comet_blast
57639f2acf86d7037f0c89e64	items/images/en/thumbs/nebula_bore.fd708a494dcd2f8a9e8fc0b16a5814d7d.128x128.png	Nebula Bore	nebula_bore
57639ee2cf86d7037f0c89e67	sub_icons/weapon/generic_blade_128x128.png	Kasses Blade	kasses_blade

BACKEND РОЗРОБЛНОЇ СТОРІНКИ

ВИСНОВКИ

- Проаналізовано обрану предметну область, визначено вимоги к завданню.
- Використано для дослідження гру «Warframe».
- Обрано технології та середовище для розробки.
- Створено адаптивну платформу інтерфейсу.
- Обрано модель штучного інтелекту яка буде давати рекомендації
- Виконано демонстрацію PoC «Proof of concept» працездатності адаптивної платформи

ВИСНОВКИ

- Проаналізовано обрану предметну область, визначено вимоги к завданню.
- Використано для дослідження гру «Warframe».
- Обрано технології та середовище для розробки.
- Створено адаптивну платформу інтерфейсу.
- Обрано модель штучного інтелекту яка буде давати рекомендації
- Виконано демонстрацію PoC «Proof of concept» працездатності адаптивної платформи