Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ВЕБДОСТУПНІСТЬ: АДАПТАЦІЯ ВЕБСАЙТІВ ДЛЯ ЛЮДЕЙ ІЗ ОБМЕЖЕНИМИ МОЖЛИВОСТЯМИ

Текстова частина до курсової роботи
за спеціальністю "Комп'ютерні науки" 122

Керівник курсової роботи
д-р. т. н., доц. Глибовець А.М.
_____
(підпис)
"____" _____ 2021 р.

Виконала студентка КН-4
Закала М.В.
"____" _____ 2021 р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри мережних технологій,
проф., доктор фіз.-мат. наук
_____Г. І. Малашонок
(підпис)
„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студентці 4 року навчання БП "Комп’ютерні науки"
Закалі Марії Володимирівні
на тему:
Вебдоступність: адаптація вебсайтів для людей із обмеженими можливостями

Зміст ТЧ до курсової роботи:
Індивідуальне завдання
Вступ
1. Аналіз предметної області. Огляд вебдоступності
2. Огляд існуючих стандартів вебдоступності та проблем, які вони покликані вирішити
3. Огляд найкращих практик вебдоступності
4. Адаптація компонента Flexmonster Pivot Table & Charts до потреб людей з обмеженими можливостями
Висновки
Список використаних джерел
Додатки (за необхідністю)

Дата видачі: „___” _____ 2020 р.

Керівник: д-р. т. н., доц. Глибовець А.М._____
(підпис)

Завдання отримала _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	15.10.2020	
2.	Огляд літератури за темою роботи.	16.11.2020	
3.	Дослідження існуючих стандартів вебдоступності та проблем, які вони покликані вирішити.	28.12.2020	
4.	Дослідження найкращих практик вебдоступності.	25.01.2021	
5.	Адаптація JavaScript компоненту Flexmonster Pivot Table & Charts до потреб людей з обмеженими можливостями.	01.03.2021	
6.	Написання текстової частини роботи.	22.03.2021	
7.	Створення презентації.	29.03.2021	
8.	Корегування роботи згідно з результатами перевірки роботи науковим керівником.	05.04.2021	
9.	Остаточне оформлення пояснювальної роботи та презентації.	09.04.2021	
10.	Подання роботи на кафедру для перевірки на плагіат.	12.04.2021	
11.	Захист курсової роботи.	Кінець квітня	

Студент _____

Керівник _____

"_____" _____

ЗМІСТ

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

W3C — World Wide Web Consortium;

WAI — Web Accessibility Initiative;

WYSIWYG — What You See Is What You Get;

HTML — HyperText Markup Language;

CMS — Content Management System;

LMS — Learning Management System;

SVG — Scalable Vector Graphics;

ICT — Information and Communications Technology;

API — Application Programming Interface;

CSS — Cascading Style Sheets;

DOM — Document Object Model;

OLAP — Online Analytical Processing.

АНОТАЦІЯ

У даній роботі обґрунтовується важливість вебдоступності, а також розглядаються сучасні стандарти вебдоступності та найкращі практики, використання яких забезпечить доступність вебконтенту для людей з особливими потребами. Аналізується вебдоступність JavaScript-компонента Flexmonster Pivot Table & Charts та проводиться його адаптація до потреб людей з обмеженими можливостями.

ВСТУП

За приблизними підрахунками, 60% населення нашої планети має доступ до мережі Інтернет [1]. Серед них є певна група людей, що через фізичні чи соціально-економічні обмеження можуть не мати можливості вільно та безперешкодно користуватися мережею Інтернет. Прикладами таких обмежень є вади зору чи низька пропускна здатність мережі користувача.

У випадку фізичних обмежень, на допомогу користувачам приходять асистивні технології. Наприклад:

- читачі екрана — спеціальне програмне забезпечення, що обробляє вміст браузера та подає його у зрозумілій користувачу формі. Наприклад, озвучує чи конвертує у шрифт Брайля;
- розпізнавання голосу — можливість пристрою розпізнавати та сприймати голосові команди користувача;
- віртуальна клавіатура — клавіатура, що відображається на екрані. Нею можна керувати мишкою, джойстиком та іншими маніпуляторами.

Якщо вебсайт чи вебзастосунок спроєктований та розроблений з урахуванням потреб людей з обмеженими можливостями, вони зможуть без проблем використовувати його з асистивними технологіями чи без них.

Втім, на даний момент більшість вебсайтів є малопридатною для людей з обмеженими можливостями. Це означає, що вони або взагалі не можуть користуватися певним сайтом, або роблять це з великими зусиллями навіть з допомогою асистивних технологій.

Отже, якщо розробники та організації прагнуть створювати якісні вебсайти для різних людей, вони повинні звертати особливу увагу на доступність своїх продуктів.

Доступність вебресурсів, або вебдоступність, означає, що вебсайт чи застосунок розробляється з урахуванням того, що ним можуть користуватися люди з обмеженими можливостями. Якщо такі користувачі зможуть самостійно

шукати, сприймати та розуміти інформацію, користуючись вебсайтом чи застосунком, то його можна вважати доступним.

Звичайно, потрібно думати про те, як зробити продукт доступним, на найпершому етапі його створення. Але оскільки наразі багато вже існуючих вебсайтів не є зручними для людей з обмеженими можливостями, гостро постає питання про їх адаптацію для таких користувачів. Отже, обрана **тема** роботи **є актуальною**.

Об'єкт дослідження — вебдоступність та основні стандарти вебдоступності.

Предмет дослідження — доступність JavaScript-компонента Flexmonster Pivot Table & Charts та його адаптація до потреб людей з обмеженими можливостями.

Метою курсової роботи є вивчення основних стандартів та найкращих практик вебдоступності, а також впровадження цих практик на вебресурсах для забезпечення їх доступності. Для досягнення мети роботи поставлено такі завдання:

- проаналізувати сучасний стан вебсайтів у контексті вебдоступності;
- ознайомитися з вадами здоров'я, які мають вплив на можливість безперешкодно користуватися мережею Інтернет;
- дослідити існуючі стандарти та найкращі практики у сфері вебдоступності;
- обрати та запровадити певні найкращі практики у JavaScript-компоненті Flexmonster Pivot Table & Charts та забезпечити його доступність для людей з особливими потребами.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ОГЛЯД ВЕБДОСТУПНОСТІ

1.1. Обґрунтування важливості доступних сайтів для усіх верств населення

За даними Всесвітньої організації охорони здоров'я, на кінець 2020 року понад один мільярд людей по всьому світі мали певний тип інвалідності [2]. Це відповідає приблизно 15% усього населення Землі, і це число продовжує збільшуватися. Багато з цих людей не можуть користуватися мережею Інтернет без асистивних технологій.

Щодо України, на початок 2020 року кількість людей з інвалідністю становила 2 703 006 осіб [3].

Ці числа показують, що прошарок населення, який потребує доступного вебу, достатньо великий. Ігнорувати ці потреби означало б дискримінувати цілу групу людей, яка має право шукати і переглядати інформацію в мережі Інтернет.

Однак це не єдина причина розробляти доступні вебсайти. Розгляньмо кілька ситуацій, які можуть статися у повсякденному житті:

- заклопотана мама, тримаючи в одній руці дитину, іншою рукою намагається швидко знайти потрібну їй інформацію в мережі Інтернет зі свого смартфона;
- під час роботи в чоловіка несподівано ламається комп'ютерна миша. Поки він чекає доставки нового пристрою, йому потрібно знайти інший шлях взаємодії з комп'ютером;
- стоячи посеред вулиці в сонячний день, група підлітків обирає в мережі Інтернет локації, які вони хочуть відвідати.

В усіх наведених ситуаціях людям було б зручно користуватися саме доступними вебсайтами: мамі б стала в нагоді адаптивність до мобільних пристроїв і можливість навігації однією рукою чи голосового вводу; чоловік зміг би виконувати потрібні йому дії, використовуючи так звані "гарячі клавіші" на

клавіатурі; підлітки легко б прочитали зі свого екрана текст навіть у сонячний день, якщо контраст тексту та його фону достатній.

Отже, з доступних вебсайтів можуть скористати не лише люди з інвалідністю, а і переважна більшість користувачів мережі Інтернет [4], а саме:

- користувачі смартфонів, розумних годинників, телевізорів з технологією Smart TV та інших пристроїв, які мають малий екран або вимагають специфічних методів вводу даних;
- похилі віком люди, чия здатність користуватися електронними пристроями змінюється через вік;
- люди з тимчасово обмеженими можливостями (як через стан здоров'я, так і через проблеми з технікою). Наприклад, через поламану руку чи непрацюючу клавіатуру;
- користувачі в певній ситуації, що обмежує їх у користуванні мережею Інтернет. Наприклад, яскраве світло на вулиці чи перебування в людному місці, де неможливо прослухати аудіо;
- користувачі, чиє перебування у мережі Інтернет обмежене повільним, лімітованим чи дорогим трафіком.

1.2. Аналіз стану сучасних вебсайтів у контексті вебдоступності

Хоча таке поняття як доступність Інтернет-ресурсів існує вже давно, і останнім часом на нього почали звертати особливо багато уваги, все ж більшість сайтів у мережі не можуть називатися доступними [5].

Так, деякі користувачі з вадами зору відмічають, що стикаються з труднощами у користуванні вебсайтами майже в 90% випадків [6]. Аналіз доступності майже мільйона найпопулярніших сайтів, проведений ресурсом WebAIM у лютому 2020 року, показав, що 98.1% домашніх сторінок вебсайтів з різних причин не є доступними [7]:

- 86.3% сторінок містили текст, що погано контрастує з фоном, через що важко читається;

- на 66% сторінок були зображення, що не мали альтернативного тексту. Це означає, що якби зображення не завантажилось чи користувач використовував програму-читач екрана, було б незрозуміло, яке саме зображення присутнє на сторінці і яку інформацію воно передає;
- 53.8% сторінок містили в собі форми з полями вводу, що не мали підписів. Якщо таку форму заповнювала б людина, що користується програмою-читачем екрана, вона не змогла б зрозуміти, яку інформацію потрібно ввести в певному полі вводу;
- на 28.0% сторінок не була вказана мова документу (атрибут lang в тезі <html>). Це ускладнює браузерам та асистивним технологіям обробку контенту сторінки.

Вищенаведені недоліки можуть створити людям з особливими потребами серйозні перешкоди на шляху до вільного користування сайтом. Чи будуть користувачі затримуватися на таких ресурсах?

Згідно з опитуванням The Click-Away Pound Report 2019 [8], 69% респондентів з обмеженими можливостями залишать сайт, яким незручно користуватися. 75% респондентів вважають, що доступність онлайн-магазинів для них важливіше за ціни, які вони пропонують.

За даними опитування, приблизні втрати від того, що люди йдуть з онлайн-магазину через його "недоступність", у 2019 році склали більше 17 мільярдів фунтів стерлінгів.

Отже, якщо вебсайт не є доступним, потенційно він може втратити досить велику кількість відвідувачів. Якщо це сайт комерційної організації, разом з відвідувачами він втрачає ще й можливість отримати фінансову вигоду. І, що найважливіше, через негативну реакцію користувачів на свій сайт компанія може набути репутацію такої, що не турбується про потреби своїх клієнтів.

Окрім збільшення аудиторії сайту та фінансової вигоди, впровадження вебдоступності матиме ще кілька позитивних наслідків:

- репутація бренду, що цінує та поважає своїх користувачів. Люди діляться не лише поганими, але й хорошими враженнями. Якщо вебсайт

компанії буде зручним у використанні і враховуватиме потреби усіх своїх клієнтів, це не залишиться непоміченим;

- краща видача сайту в пошукових системах (SEO). Коли пошукові системи відбирають результати на запит користувача, вони враховують в тому числі коректність HTML-верстки на сайті, наявність структури контенту (наприклад, з використанням заголовків <h1>-<h6>) та інші технічні моменти, які мають пряме відношення і до вебдоступності також. Доступний сайт буде краще сприйматися не лише асистивними технологіями типу читачів екрана, але й пошуковими системами;
- уникнення судових справ. Уряд багатьох розвинених країн почав впроваджувати вимоги до доступності вебсайтів на законодавчому рівні. Недотримання цих вимог вважається дискримінацією людей з обмеженими можливостями. Якщо такий вебсайт належить компанії, що здійснює свою діяльність публічно, і якийсь користувач не зміг скористатися її послугами в мережі Інтернет через труднощі з доступністю, компанію можна буде притягнути до відповідальності.

1.3. Висновки

Отже, створення доступних вебсайтів принесе користь не лише людям з обмеженими можливостями, а і тим, хто користується пристроями з маленькими екранами чи специфічними методами вводу. Також доступний сайт буде зручним для тих, хто стикається з труднощами через похилий вік, має тимчасово обмежені можливості або проблеми з Інтернет-трафіком.

Наразі переважна частина вебсайтів у мережі Інтернет має ті чи інші проблеми з доступністю, що створюють перешкоди для користувачів з особливими потребами.

Адаптація вебсайту до потреб користувачів з обмеженими можливостями допоможе зберегти та збільшити аудиторію, а також покращить його видачу

пошуковими системами. Якщо доступний вебсайт належить комерційній організації, він зміцнюватиме її репутацію та приноситиме фінансову вигоду.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ СТАНДАРТІВ ВЕБДОСТУПНОСТІ ТА ПРОБЛЕМ, ЯКІ ВОНИ ПОКЛИКАНІ ВИРІШИТИ

2.1. Огляд відомих вад здоров'я та труднощів у користуванні мережею Інтернет, які виникають внаслідок них

Для того, щоб зробити інтерфейс вебсайту доступним для людей з обмеженими можливостями, потрібно розуміти, з якими труднощами стикаються ці люди у мережі Інтернет [9].

2.1.1. Вади зору

За даними Всесвітньої організації охорони здоров'я, у світі близько 2.2 мільярди людей мають ті чи інші вади зору [10]. Приблизно половина з цих людей має помірні чи виражені вади зору.

2.1.1.1. Сліпота

Сліпота — це повна або майже повна втрата зору.

Щоб користуватися мережею Інтернет, сліпі люди можуть використовувати такі асистивні технології, як читачі екрана чи оновлювані дисплеї Брайля.

Труднощі, з якими можуть стикатися сліпі люди у мережі Інтернет:

- зображення, які не мають альтернативного тексту. В такому разі, програма-читач екрана не зможе зчитати зображення чи його опис, і сліпа людина не зрозуміє, що саме зображено на картинці;
- складні зображення, такі як графіки чи діаграми, яким бракує опису. Аналогічно до попереднього пункту — програма-читач екрана не зможе передати інформацію з графіку чи діаграми, якщо вона додатково не описана;

- відео, що не має опису в аудіо- чи текстовому форматі. Сліпа людина не зможе зрозуміти, що саме показано на відео, без аудіо супроводу чи текстового опису, який може бути озвучений програмою-читачем екрана;
- таблиці, інформація в яких не матиме сенсу при зачитуванні програмою-читачем екрана таблиці клітинка за клітинкою;
- елементи вводу форм, які не мають зрозумілих підписів чи які йдуть в неправильному порядку за умови навігації з клавіатури;
- браузери та програмне забезпечення, чий функціонал не може повною мірою використовуватися за допомогою клавіатурних скорочень.

2.1.1.2. Поганий зір

Термін "поганий зір" включає в себе такі вади зору, як знижена гострота зору, втрата периферичного зору (тунельний зір), втрата центрального зору, затуманення зору.

Люди, що мають поганий зір, зазвичай не використовують програми-читачі екрана, але можуть використовувати монітори з великим розміром екрана чи збільшувати системні шрифти та зображення. Ще одним методом збільшення вмісту екрана є використання так званої екранної лупи.

Щоб краще сприймати текст, люди з низьким зором можуть потребувати особливих налаштувань кольорів тексту та фону, наприклад, жовтий текст на чорному фоні. Вони також можуть використовувати для певних сторінок власні стилі, якщо вважають їх більш зручними для себе.

Труднощі, з якими можуть стикатися люди з поганим зором у мережі Інтернет:

- вебсторінки з абсолютним розміром шрифтів, який важко збільшити;
- вебсторінки з погано продуманим інтерфейсом користувача, який ускладнює навігацію по сторінці, коли використовується екранна лупа;

- браузери, які не дозволяють користувачам перевизначити стилі певної вебсторінки;
- низький контраст кольорів, що ускладнює сприйняття тексту;
- замалі іконки, навігаційні елементи та елементи керування сторінкою. У випадку малих розмірів зазначених складових сторінки, людині з поганим зором може бути важко розібрати, що означає певна іконка, чи влучити курсором на потрібний елемент управління (наприклад, кнопку).

2.1.1.3. Колірна сліпота

Колірна сліпота (також відома як дальтонізм) виражається в нездатності людини розрізняти певні кольори.

Люди з колірною сліпотою можуть заміняти стилі певної сторінки на свої, якщо вважають їх більш зручними для себе в контексті вдалого підбору кольорів.

Труднощі, з якими можуть стикатися люди з дальтонізмом у мережі Інтернет:

- інформація, яка передається лише з допомогою певного кольору (наприклад, після неправильно введеного пароля під час авторизації, поле для вводу пароля стає червоним без будь-якого пояснювального тексту);
- текст, який важко сприймати через непідходящий контраст з фоном чи навколишніми елементами;
- браузери, що не дозволяють користувачам перевизначити стилі певної вебсторінки.

2.1.2. Вади слуху

За даними Всесвітньої організації охорони здоров'я, приблизно півтора мільярда людей у всьому світі мають вади слуху, з них близько 430 мільйонів мають помірні чи виражені вади [11].

2.1.2.1. Глухота

Глухотою є повна втрата слуху в обох вухах.

Під час перебування у вебі, глухі люди потребують текстового опису до аудіоконтенту.

Труднощі, з якими можуть стикатися люди з глухотою у мережі Інтернет:

- відсутність текстового опису до аудіоконтенту;
- сторінки, які містять лише текст без інших візуальних складових (наприклад, зображень). У повсякденному житті глухі люди зазвичай послуговуються жестовими мовами, тому їм може бути важко сприймати інформацію, подану таким чином.

2.1.2.2. Часткова втрата слуху

Частковою втратою слуху вважаються незначні чи помірні вади слуху.

Під час перебування у вебі, люди з частковою втратою слуху можуть сприймати аудіоконтент за допомогою текстового опису чи підвищувати гучність динаміків.

Труднощі, з якими можуть стикатися люди з частковою втратою слуху у мережі Інтернет:

- відсутність текстового опису до аудіоконтенту.

2.1.3. Моторні порушення

До моторних порушень можна віднести:

- слабкість м'язів;
- порушення контролю м'язів (наприклад, мимовільні скорочення м'язів, погана координація, параліч);
- хвороби суставів;
- втрата кінцівок.

Усі вищеперераховані порушення впливають на те, як людина володіє своїми руками та іншими частинами тіла. Так, деякі можуть послуговуватися клавіатурою як основним інструментом для навігації, іншим же потрібні будуть альтернативні методи керування, наприклад, голосовий.

Труднощі, з якими можуть стикатися люди з моторними порушеннями у мережі Інтернет:

- браузери та програмне забезпечення, чий функціонал не може повною мірою використовуватися за допомогою клавіатурних скорочень;
- елементи форми, які йдуть в неправильному порядку за умови навігації з клавіатури;
- інтерактивні форми, що мають обмежений час відповіді.

2.1.4. Когнітивні та неврологічні вади

2.1.4.1. Розлади навчання

Розлади навчання можуть впливати на здатність людини сприймати та запам'ятовувати інформацію, а також концентруватися на конкретній задачі. Залежно від типу розладу, людина може краще сприймати інформацію одним органом відчуття і гірше іншим. Наприклад, людині може бути зручно слухати розповідь, але незручно читати її.

Під час перебування у вебі, люди з розладами навчання потребують, щоб інформація була подана кількома шляхами. Наприклад, людина, якій важко

сприймати інформацію візуально, може захотіти конвертувати її в аудіо для кращого розуміння. А той, хто погано сприймає на слух, може звернутися до текстового опису аудіозапису. Людина, яка легко відволікається, може скористатися можливістю відключити анімацію на сайті, щоб зосередитися на контенті.

Труднощі, з якими можуть стикатися люди з розладами навчання у мережі Інтернет:

- відсутність альтернативних методів подачі інформації. Наприклад, візуальні елементи без текстового опису, який можна конвертувати в аудіо, і навпаки — аудіозаписи без опису;
- наявність візуальних чи аудіоелементів, які відволікають увагу та які не можна легко вимкнути;
- погана організація вебсайту, що ускладнює навігацію по ньому та сприйняття його вмісту.

2.1.4.2. Порушення інтелектуального розвитку

Такі люди можуть повільніше навчатися чи мати складнощі з розумінням складних речей.

Під час перебування у вебі, користувачі з порушеннями інтелектуального розвитку можуть потребувати більше часу на вивчення вмісту вебсайту. Також, ймовірно, вони краще розумітимуть інформацію, подану графічно (наприклад, через зображення, графіки та діаграми), та краще сприйматимуть тексти, написані простою, зрозумілою мовою.

Труднощі, з якими можуть стикатися люди з порушенням інтелектуального розвитку у мережі Інтернет:

- погана організація вебсайту, що ускладнює навігацію по ньому та сприйняття його вмісту;

- тексти, написані з використанням невиправдано складних мовних конструкцій, які ускладнюють його розуміння;
- недостатня кількість графічних елементів на вебсайтах.

2.1.4.3. Порушення пам'яті

Під час перебування у вебі, користувачі з порушеннями пам'яті покладаються на зрозумілу, послідовну та інтуїтивну навігацію на вебсайті. На такому вебсайті легше орієнтуватись, і навіть якщо людина забуде, де знаходиться чи куди веде той чи інший навігаційний елемент, вона зможе легко розібратися в цьому.

Труднощі, з якими можуть стикатися люди з порушенням пам'яті у мережі Інтернет:

- погана організація вебсайту, що ускладнює навігацію по ньому та сприйняття його вмісту.

2.1.4.4. Епілепсія

Деяким людям з епілепсією можуть нашкодити миготливі візуальні елементи чи аудіозаписи, які мають певну частоту.

Щоб комфортно користуватися мережею Інтернет, люди з епілепсією повинні мати можливість відключити анімацію, миготливий текст чи аудіо. Уникнення таких частот може запобігти провокуванню нападу епілепсії.

Труднощі, з якими можуть стикатися люди з епілепсією у мережі Інтернет:

- відео- та аудіочастоти, які можуть спровокувати напад епілепсії.

2.1.4.5. Психічні розлади

Людям з ментальними чи емоційними розладами може бути важко концентрувати свою увагу.

Щоб комфортно користуватися мережею Інтернет, такі люди повинні мати можливість відключати візуальні чи аудіоелементи, що відволікають увагу.

Труднощі, з якими можуть стикатися люди з психічними розладами у мережі Інтернет:

- погана організація вебсайту, що ускладнює навігацію по ньому та сприйняття його вмісту;
- наявність візуальних чи аудіоелементів, які відволікають увагу та які не можна легко вимкнути.

2.2. Огляд існуючих стандартів вебдоступності

2.2.1. Web Content Accessibility Guidelines (WCAG)

Технічний стандарт WCAG [12] є одним із стандартів вебдоступності, розроблених в рамках Ініціативи вебдоступності (WAI) міжнародною організацією W3C, яка розробляє та впроваджує технічні стандарти для Всесвітнього вебу [13]. Дана ініціатива покликана зробити Всесвітній веб зручнішим для людей з особливими потребами [14].

Стандарт WCAG був створений з метою стати єдиним стандартом доступності вебконтенту, який бере до уваги потреби людей, організацій та урядів по цілому світі.

Даний стандарт пояснює, як зробити вміст вебсайтів більш доступним для людей з особливими потребами. Термін "вебконтент" означає інформацію, що міститься на вебсторінці чи застосунку, включно з:

- текстами, зображеннями та музикою;
- кодом та розміткою, які визначають структуру, зовнішній вигляд сторінки і т. д.

Стандарт WCAG призначений для:

- людей, які створюють контент для вебу (контент-менеджери вебсайтів, дизайнери інтерфейсів користувача і т. д.);

- розробників, які створюють програмне забезпечення для людей, які створюватимуть контент для вебу;
- розробників, які створюють програмне забезпечення для перевірки вебдоступності застосунків;
- всіх інших, кому потрібен стандарт вебдоступності, включно з доступністю мобільних застосунків.

Стандарт WCAG містить 12-13 (залежно від версії) рекомендацій, організованих за чотирма основними принципами, яким має відповідати доступний вебсайт: помітність, керованість, зрозумілість та надійність. Кожна рекомендація має певні "критерії успіху", за якими можна визначити, чи вона виконана. Критерії мають один з трьох рівнів відповідності: A (найнижчий), AA (середній), AAA (найвищий).

Наразі стандарт WCAG має дві опублікованих версії:

- WCAG 2.0, опублікований 11 грудня 2008 року [15];
- WCAG 2.1, опублікований 5 червня 2018 року [16]. Він містить усі ті самі вимоги ("критерії успіху"), що і версія 2.0, а також нові, додані у версії 2.1.

Версія 2.2 запланована до публікації у 2021 році.

Версії стандарту є зворотно сумісними. Це означає, що сайт, який відповідає стандарту WCAG 2.1, відповідає також і стандарту WCAG 2.0. Аналогічно, вебсайт, який буде доступним за версією 2.2, відповідатиме також і версії 2.1.

Обидві опубліковані версії стандарту є чинними. Хоча поява нової версії стандарту не робить інші версії застарілими, W3C заохочує розробників орієнтуватися на останню версію стандарту під час розробки. Так створений продукт відповідатиме найсучаснішим стандартам та вимогам вебдоступності.

2.2.2. Authoring Tools Accessibility Guidelines (ATAG)

Технічний стандарт ATAG [17] належить до серії стандартів вебдоступності, розроблених в рамках Ініціативи вебдоступності (WAI) міжнародною організацією W3C.

Стандарт ATAG розповсюджується на програмне забезпечення та сервіси, які використовуються розробниками, дизайнерами, письменниками та іншими людьми для створення вебконтенту, а саме:

- інструменти для створення вебсторінок. Наприклад, WYSIWYG редактори HTML-коду;
- програмне забезпечення для створення вебсайтів. Наприклад, CMS чи LMS;
- програмне забезпечення, що здійснює конвертацію файлів у формати для відображення у вебі (наприклад, HTML чи EPUB);
- інструменти для створення та роботи з мультимедіа;
- онлайн-форуми, соціальні мережі та вебсайти, які дозволяють своїм користувачам створювати контент (наприклад, блоги та вікі) або ділитися фотографіями;
- інші інструменти, які підпадають під визначення "authoring tools" [18].

Стандарт ATAG має дві версії: 1.0 та 2.0. Стандарт 1.0 був випущений в лютому 2000 року і, хоч і є коректним, вже вважається застарілим. Рекомендованим стандартом є ATAG 2.0, оскільки він є завершеним та пройшов багато випробувань, в тому числі і на реальних проєктах.

ATAG 2.0 має дві основні частини:

- частина А пояснює, як розробляти інструменти, з допомогою яких люди з обмеженими можливостями матимуть змогу створювати вебконтент;
- частина В покликана допомогти авторам створювати доступний контент для вебу, а саме уможливити, підтримувати та заохочувати створення контенту, який би відповідав стандарту WCAG.

Як і стандарт WCAG, ATAG 2.0 також містить рекомендації, кожна з яких має свої критерії з трьома рівнями відповідності: A (найнижчий), AA (середній), AAA (найвищий).

2.2.3. User Agent Accessibility Guidelines (UAAG)

Технічний стандарт UAAG [19] належить до серії стандартів вебдоступності, розроблених в рамках Ініціативи вебдоступності (WAI) міжнародною організацією W3C.

Даний стандарт надає рекомендації щодо того, як створювати доступними для людей з обмеженими можливостями такі програми:

- браузери;
- розширення для браузерів;
- медіаплеєри;
- інші програми, що здійснюють візуалізацію вебконтенту.

Браузер, який відповідає стандарту UAAG, буде покращувати доступність вебконтенту завдяки власному інтерфейсу та здатності взаємодіяти з іншими технологіями, в тому числі і асистивними.

Стандарт UAAG має дві версії: 1.0 та 2.0. Версія 2.0 наразі є рекомендованою, оскільки вона орієнтована саме на сучасні браузери.

2.2.4. WAI-ARIA

Технічний стандарт WAI-ARIA (Accessible Rich Internet Applications Suite) [20] був розроблений в рамках Ініціативи вебдоступності (WAI) міжнародною організацією W3C.

Він визначає, як зробити вебконтент та вебзастосунки доступними для людей з особливими потребами. Хоча мета цього стандарту і перегукується з метою стандарту WCAG, він спрямований більшою мірою на динамічний

контент та нестандартні елементи керування, створені з допомогою HTML, JavaScript та інших подібних технологій.

На відміну від попередніх стандартів, які надавали лише рекомендації щодо досягнення вебдоступності, стандарт WAI-ARIA надає фреймворк для визначення ролей та властивостей різних елементів вебсторінки, а також взаємозв'язку та поточного стану цих елементів.

Правильне використання цього фреймворку гарантує, що читачі екрана та інші асистивні технології матимуть можливість коректно обробити вебсторінку, а це означає, що вона буде доступною для людини з особливими потребами. Отже, цей фреймворк є одним з основних інструментів у створенні вебсторінок, що відповідають стандарту WCAG.

Фреймворк WAI-ARIA надає розробникам можливість використовувати:

- ролі для позначення типу певного віджета. Наприклад, "slider", "button", "checkbox";
- ролі для позначення структурних елементів вебсторінки, таких як заголовки, зображення, чи таблиці;
- властивості для позначення стану віджетів (наприклад, "checked" для чекбоксів або "readonly" для елементів, що призначені лише для читання);
- властивості для позначення тих частин сторінки, які можуть оновлюватися в режимі реального часу.

Стандарт WAI-ARIA має дві опублікованих версії: 1.0 (20 березня 2014 року) та 1.1 (14 грудня 2017 року). У розробці також знаходиться версія 1.2.

Окрім інших оновлень, що розширять стандарт WAI-ARIA 1.1, версія 1.2 міститиме в собі визначення того, як браузери та інші інструменти повинні обробляти SVG-розмітку. Це допоможе поліпшити доступність SVG-елементів, адже зараз досить важко зробити їх доступними для людей з особливими потребами.

2.2.5. Section 508

Повна назва стандарту — Section 508 Amendment to the Rehabilitation Act of 1973 [21].

Section 508 — це американський стандарт, що має на меті забезпечити доступність не лише вебу, а й програмного забезпечення, документації до нього і т. д. Безпосередньо вебдоступності стосується параграф 1194.22 стандарту: Web-based intranet and internet information and applications [22].

Дана частина має 16 вимог, 11 з яких є переформульованими рекомендаціями із стандарту WCAG. 5 інших вимог специфічні стандарту Section 508 [23].

Варто зазначити, що стандарт Section 508 є офіційно прийнятим у США і має юридичну силу, тож його обов'язково повинні дотримуватися розробники, які створюють програмне забезпечення та сервіси для бізнесів, що працюють на загал в одній з цих категорій [24]:

- житло;
- їжа та напої;
- розваги та виставки;
- публічні збори;
- продажі та оренда;
- сфера обслуговування;
- громадський транспорт;
- публічний показ;
- рекреація;
- освіта;
- соціальні служби;
- фізичні вправи або відпочинок.

Порушення стандарту Section 508 може призвести до того, що власників бізнесу притягнуть до відповідальності.

2.2.6. Короткий огляд інших стандартів вебдоступності

Варто зазначити, що більшість стандартів вебдоступності, які є офіційно прийнятими у різних країнах світу, є повними чи частковими аналогами стандарту WCAG. Оскільки сам стандарт WCAG не має юридичної сили і носить лише рекомендаційний характер, уряди країн створюють власні стандарти, що містять рекомендації з нього.

2.2.6.1. Канада — Standard on Web Accessibility

Даний стандарт є офіційною адаптацією стандарту WCAG [25]. Відповідно до цього стандарту, усі офіційні вебсайти та вебзастосунки уряду Канади повинні відповідати критеріям WCAG на рівні, не меншому від AA.

2.2.6.2. Європа — EN 301 549

Європейський стандарт цифрової доступності, який містить вимоги до державних закупівель ICT-продуктів та сервісів у Європі [26]. Є офіційним аналогом стандарту WCAG.

2.2.6.3. ISO 30071-1 Digital Accessibility Standard

Самостійний міжнародний стандарт, який містить рекомендації з впровадження та використання інклюзивних практик в організаціях, а також під час створення ICT-продуктів [27].

2.2.6.4. Україна

В Україні поки немає офіційно прийнятих стандартів, які вимагали б створення вебсайтів, доступних для людей з особливими потребами. Втім, перші кроки до цього вже було зроблено.

Постанова Кабінету Міністрів України від 12 червня 2019 року №493 "Про внесення змін до деяких постанов Кабінету Міністрів України щодо функціонування офіційних веб-сайтів органів виконавчої влади" [28] встановила вимоги до державних вебсайтів. Відповідно до цих вимог, усі офіційні вебсайти органів виконавчої влади повинні відповідати критеріям зі стандарту WCAG з рівнем, не нижчому від AA.

2.3. Висновки

Отже, усі вади здоров'я можна умовно поділити на такі групи: вади зору, слуху, фізичні, когнітивні та неврологічні вади. Вони по-різному впливають на те, як людина користуватиметься мережею Інтернет: наприклад, одні користувачі не можуть обійтися без асистивних технологій, інші потребують простої та послідовної структури вебсайту, а також контент, який легко сприймати.

Для того, щоб вебсайти відповідали потребам людей з вадами здоров'я, було створено такі стандарти веб доступності, як WCAG, ATAG, UAAG, WAI-ARIA, Section 508 та інші. Section 508 відноситься до стандартів, затверджених на державному рівні. Невідповідність вебсайтів таким стандартам може призвести до того, що його власники будуть притягнуті до відповідальності.

З рекомендацій, що містяться у згаданих стандартах, можна зробити висновок, що доступність вебсайту для людей з особливими потребами залежить від кожного, хто працює над вебсайтом: як від розробника, що відповідає за технічну складову вебсайту, так і від людини, яка створює безпосередньо контент вебсайту.

РОЗДІЛ 3. НАЙКРАЩІ ПРАКТИКИ ВЕБДОСТУПНОСТІ

3.1. Основні рекомендації зі стандарту WCAG

Для огляду найкращих практик вебдоступності було обрано стандарт WCAG, оскільки рекомендації з нього можуть бути застосовані до JavaScript-компонента Flexmonster Pivot Table & Charts [29]. Отже, розуміння цих рекомендацій стане в нагоді під час виконання практичної частини роботи. До того ж, саме цей стандарт став основою для більшості офіційно прийнятих стандартів вебдоступності. Отже, якщо розробники дотримуватимуться рекомендацій з WCAG, їх продукт буде відповідати також багатьом офіційним стандартам.

Як згадувалося у минулому розділі, стандарт WCAG, залежно від версії, містить 12-13 рекомендацій. Якщо під час розробки продукту брати до уваги ці рекомендації, він буде доступним для людей з особливими потребами.

У даному розділі буде коротко розглянуто рекомендації з версії стандарту 2.1, оскільки саме вона є найостаннішою.

Усі рекомендації зі стандарту WCAG поділено за чотирма основними принципами вебдоступності: помітність, керованість, зрозумілість та надійність.

3.1.1. Помітність

Згідно з цим принципом, елементи інтерфейсу користувача та інформація, подана на вебсайті, повинна легко сприйматися усіма користувачами. Наприклад, графіки чи зображення на вебсторінці повинні мати текстовий опис, щоб з поданою в них інформацією могла ознайомитися людина, яка використовує програму-читач екрана.

Під даний принцип підпадають наступні рекомендації:

1. **текстові альтернативи.** Варто надавати текстовий опис чи альтернативу нетекстовому контенту, до якого належать зображення,

відео, аудіо, графіки і т. д. У такому разі, користувачі за потреби матимуть можливість змінити подачу інформації на таку, яка буде зручною саме їм. Наприклад, вони зможуть конвертувати текст у мову чи шрифт Брайля;

2. **медіа, що має певну тривалість.** Це поняття є вужчим за поняття нетекстового контенту та включає в себе аудіозаписи, відео, фільми і т. д. Відповідно до цієї рекомендації, медіа, що має певну тривалість, повинно мати альтернативу у вигляді тексту, аудіо чи іншого типу медіа. Це не стосується того медіа, що було додане на вебсторінку в якості альтернативи тексту;
3. **адаптивність контенту.** Варто створювати контент так, щоб його сенс та структура не втрачалися при зміні способу відображення контенту (наприклад, при зміні орієнтації екрана мобільного пристрою з альбомної на книжну);
4. **розбірливий контент.** Контент повинен бути таким, щоб його можна було легко побачити чи почути. Наприклад, текст повинен добре контрастувати з фоном.

3.1.2. Керованість

Відповідно до цього принципу, будь-який користувач повинен мати можливість взаємодіяти з усіма елементами інтерфейсу користувача та навігації. Наприклад, цей принцип не виконуватиметься на вебсторінці, яка вимагає виключно голосового вводу даних, адже нею не зможе скористатися людина з вадами мовлення.

Під даний принцип підпадають наступні рекомендації:

1. **доступність з клавіатури.** Весь функціонал контенту повинен бути доступний за умови навігації з клавіатури;
2. **достатня кількість часу.** Користувачі повинні мати стільки часу на ознайомлення з контентом, скільки їм потрібно. Наприклад, якщо час, за

який користувач має прочитати текст, є обмеженим, користувач повинен мати можливість налаштувати це обмеження під себе.;

3. **апоплексичні удари та фізичні реакції.** Потрібно уникати контенту, якщо відомо, що він може спровокувати апоплексичний удар чи іншу фізичну реакцію. Наприклад, це стосується анімацій, які блимають частіше, ніж три рази в секунду;
4. **зрозуміла навігація.** Користувачам має бути зручно користуватися навігацією і шукати контент, а також вони повинні розуміти, де саме знаходяться відносно структури контенту;
5. **методи вводу даних.** Згідно з цією рекомендацію, потрібно намагатися створювати контент так, щоб користувачам було зручно взаємодіяти з його функціоналом через різні методи вводу (наприклад, джойстик чи інші маніпулятори).

3.1.3. Зрозумілість

Відповідно до цього принципу, подана в контенті інформація повинна бути зрозумілою користувачеві, а інтерфейс користувача — простим та інтуїтивним.

Під даний принцип підпадають наступні рекомендації:

1. **читабельність.** Контент повинен бути читабельним та зрозумілим. Для людей, що створюють контент, це означає, що варто уникати складних мовних конструкцій та рідковживаних слів, якщо в цьому немає потреби, а також використовувати просту структуру подачі інформації.

   Також важливою умовою читабельності є чітке визначення мови вебсторінки чи її частин, а також наявність механізмів, які дозволяють зрозуміти значення жаргонних слів та ідіом чи розшифрувати абревіатури. Наприклад, у випадку абревіатур можна або надати посилання на її тлумачення, або використати HTML-тег <abbr>, який дозволяє надавати пояснення абревіатур;

2. **передбачуваність.** Вебсторінки повинні мати передбачуваний інтерфейс користувача. Наприклад, по кліку на кнопку відкривається модальне вікно. Якщо користувач з допомогою клавіатури переводить фокус на цю кнопку, то модальне вікно повинно відкритися лише тоді, коли цю дію ініціює користувач (наприклад, натиснувши потрібну клавішу на клавіатурі). Якщо ж модальне вікно відкриватиметься відразу після того, як користувач переведе клавіатурний фокус на кнопку, така поведінка не буде передбаченою;
3. **допомога під час взаємодії з контентом.** Ця рекомендація націлена на те, щоб допомогти користувачам уникати помилок (наприклад, під час вводу даних) та виправляти їх. Так, підписи та вказівки до елементів вводу (скажімо, щодо очікуваного формату вхідних даних) допоможуть користувачеві зрозуміти, що саме він повинен ввести. Чіткий опис помилок дасть зрозуміти, яку неточність допустив користувач під час вводу, а з контекстною допомогою людина не загубиться у функціоналі вебсторінки.

3.1.4. Надійність

Контент є надійним, якщо його може однаково інтерпретувати велика кількість браузерів, медіаплеєрів та інших технологій, в тому числі й асистивних. Це означає, що контент повинен бути доступним користувачам як зараз, з допомогою сучасного програмного забезпечення, так і в майбутньому, коли технології будуть розвиватися.

Під даний принцип підпадають наступні рекомендації:

1. **сумісність.** Потрібно максимізувати сумісність контенту з сучасними і майбутніми технологіями, включно з асистивними. Сумісність може бути забезпечена шляхом створення коректної HTML-розмітки та використання атрибутів, що дозволяють асистивним технологіям

отримувати інформацію про ім'я, роль та значення певного елемента розмітки.

3.2. Використання фреймворку WAI-ARIA

Фреймворк WAI-ARIA надає розробникам набір додаткових HTML-атрибутів, використання яких в розмітці дозволить покращити сумісність продукту, що розробляється, з асистивними технологіями.

Варто зазначити, що атрибути WAI-ARIA ніяк не впливають на відображення вебсторінки, її структуру чи вміст. Від використання цих атрибутів залежить лише інформація, яку API браузера надає асистивним технологіям для обробки вебсторінки. Також атрибути WAI-ARIA можуть використовуватись для створення CSS-селекторів.

Усі атрибути WAI-ARIA поділяються на три типи: ролі, властивості та стани [30].

3.2.1. Ролі

Як можна здогадатися, роль елемента визначає, чим він є чи що робить. Ця інформація про елемент дасть асистивним технологіям розуміти, як саме потрібно його обробити.

Роль задається з допомогою атрибута role, наприклад, role="button".

За своїм призначенням, ролі WAI-ARIA поділяються на такі:

- **ролі для віджетів** (наприклад: button, checkbox, radio, grid, menu, тощо);
- **ролі для виділення певних навігаційних частин сторінки** (наприклад: banner, form, navigation, search, тощо);
- **ролі для позначення структури документу** (наприклад: article, heading, img, listitem, list, table, тощо);

- **ролі для позначення так званих "живих регіонів"** — тих частин сторінки, вміст яких може оновлюватись в режимі реального часу (наприклад: alert, log, timer, тощо);
- **ролі для вікон**, а саме alertdialog та dialog.

Як можна помітити, деяка кількість ролей, запропонованих WAI-ARIA, дублює семантичне значення нативних елементів HTML5 (наприклад, роль "button" та елемент <button>). Інші ж ролі описують структури, які часто зустрічаються в інтерфейсах користувача. Наприклад, банери (role="banner") та вікна пошуку (role="searchbox").

Важливо, що у випадку використання нативних HTML-елементів вказувати їх роль не обов'язково, адже усі елементи HTML5 мають відповідні ролі за замовчуванням. Зазначати ARIA-роль важливо тоді, коли потрібно створити структуру, для якої поки не існує відповідних HTML-елементів, або коли є необхідність надати властивості одного HTML-елемента іншому. Наприклад, якщо розробник хоче створити елемент , що поводитиметься як кнопка, то він має прописати відповідну роль для цього елемента: . Втім, рекомендується використовувати нативні елементи HTML завжди, коли тільки можливо, оскільки вони мають максимальну сумісність з асистивними технологіями [31].

3.2.2. Властивості

Властивості використовуються для того, щоб надати додаткову інформацію про певний HTML-елемент (наприклад, щоб зазначити, що поле обов'язково має бути заповнене, потрібно вказати атрибут aria-required="true"). Як правило, властивості елемента залишаються незмінними протягом усього періоду його існування. Цим вони відрізняються від *станів*, які можуть багато разів змінювати свої значення.

В HTML-розмітці властивості задаються як окремий атрибут з префіксом aria- та відповідним значенням.

За областю використання властивості поділяються на такі:

- **універсальні властивості.** Можуть використовуватися з усіма HTML-елементами та ARIA-ролями. Наприклад, aria-label (підпис до елемента), aria-controls (визначає елемент чи набір елементів, вміст який контролюється даним елементом), aria-describedby (визначає елемент чи набір елементів, що описують даний об'єкт, скажімо, зображення), тощо;
- **властивості віджетів.** Наприклад, aria-readonly (тільки для читання), aria-required (поле є обов'язковим для заповнення), aria-placeholder (значення-підказка у полі вводу), тощо.

  Властивості віджетів можуть використовуватися з ролями для віджетів;
- **властивості "живих регіонів".** Наприклад, aria-live (позначає, що вміст елемента буде змінюватися, та зазначає, яких саме змін варто очікувати).

  Властивості "живих регіонів" можуть бути застосовані до будь-якого елемента розмітки, але використовувати їх варто лише для тих елементів, вміст яких може динамічно змінюватися без втручання користувача;
- **властивості зв'язку.** Наприклад, зазначені вище aria-controls та aria-describedby, а також aria-rowspan (кількість рядків, яку клітинка займає в таблиці), aria-details (визначає елемент, який надає розширений опис даного об'єкта, скажімо, зображення), тощо.

  Властивості зв'язку слід використовувати для позначення зв'язку між елементами та їх взаємодії, якщо це не можна визначити візуально чи зі структури вебсторінки. Ці властивості є особливо корисними у випадках використання асистивних технологій, а також покращують навігацію з клавіатури.

3.2.3. Стани

Подібно до властивостей, стани також передають додаткову інформацію про елемент, а саме про його стан. Від властивостей стани відрізняються тим, що їх значення можуть неодноразово змінюватися. Як правило, це відбувається з використанням засобів JavaScript.

Оскільки фактично стани відрізняються від властивостей лише тим, що можуть змінювати свої значення, вони за областю використання поділяються майже на ті ж категорії, що і властивості:

- **універсальні стани.** Можуть використовуватися з усіма HTML-елементами та ARIA-ролями. Наприклад, aria-hidden (вказує, чи елемент видимий через API браузера — може використовуватися разом з CSS-властивістю hidden), aria-disabled (з елементом не можна взаємодіяти), тощо;
- **стани віджетів.** Наприклад, aria-checked (вказує, чи вибрано значення даного чекбоксу, радіо кнопки чи іншого віджета), aria-expanded (вказує, чи розгорнутий даний елемент), aria-pressed (вказує, чи натиснені кнопки перемикання), тощо. Стани віджетів можуть використовуватися з ролями для віджетів;
- **стани "живих регіонів"**, а саме aria-busy (вказує, що вміст елемента знаходиться в процесі оновлення, і асистивні технології мають дочекатися його завершення, щоб обробити цей елемент).

3.3. Висновки

У даному розділі було розглянуто найкращі практики, які слід використовувати під час розробки продукту для забезпечення його доступності, а саме рекомендації зі стандарту WCAG 2.1 та фреймворк WAI-ARIA.

Відповідно до стандарту WCAG 2.1, розроблюваний продукт повинен відповідати таким основним принципам: помітність, керованість, зрозумілість та

надійність. Створені відповідно до цих принципів рекомендації є вичерпними, тож слідування їм допоможе створити доступний продукт.

Фреймворк WAI-ARIA містить набір ролей, властивостей та станів, використання яких значно поліпшить сумісність продукту з асистивними технологіями, адже вони зможуть краще визначати роль елементів та їх взаємозв'язок, а також відслідковувати зміну стану елементів.

РОЗДІЛ 4. АДАПТАЦІЯ КОМПОНЕНТА FLEXMONSTER PIVOT TABLE & CHARTS ДО ПОТРЕБ ЛЮДЕЙ З ОБМЕЖЕНИМИ МОЖЛИВОСТЯМИ

Flexmonster Pivot Table & Charts (надалі Flexmonster) — це JavaScript-компонент, який дозволяє вбудовувати у вебзастосунки готові до використання зведені таблиці для візуалізації даних та побудови звітів.

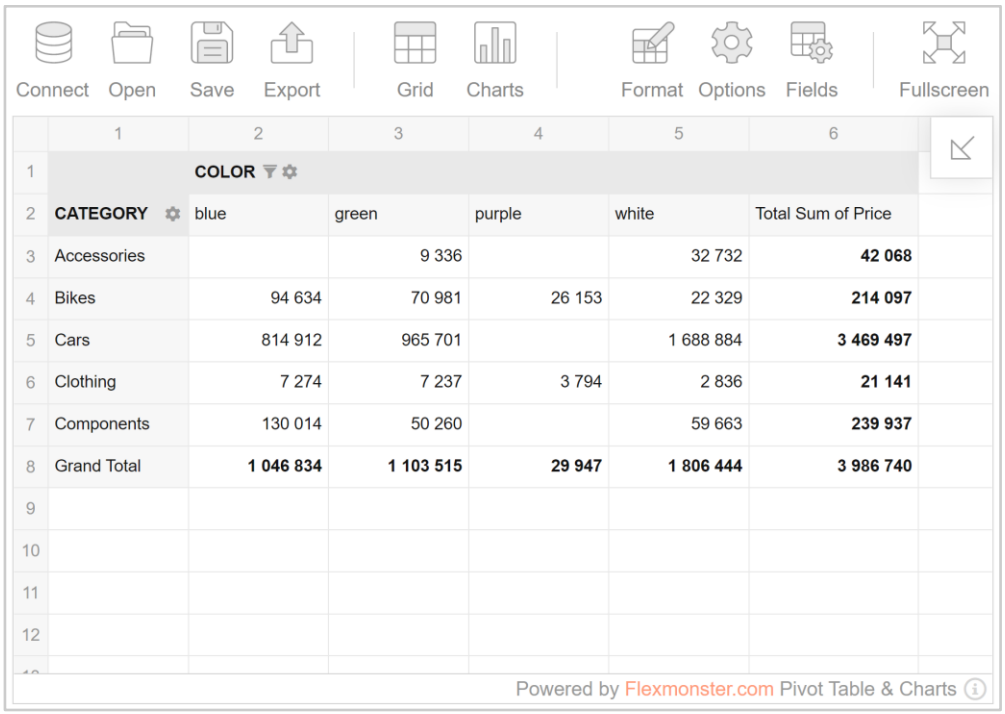


Рисунок 4.1. Інтерфейс компонента Flexmonster

Хоча Flexmonster не є самостійним вебзастосунком, він може бути складовою інших застосунків. Таким чином, його доступність є важливою для розробників, які використовують даний компонент у своїх проєктах і хочуть бути впевнені у тому, що кінцевий продукт буде доступним для всіх користувачів.

Отже, виникла необхідність у проведенні перевірки компонента Flexmonster на відповідність сучасним стандартам вебдоступності.

4.1. Тестування вебдоступності компонента Flexmonster Pivot Table & Charts

Оскільки Flexmonster не є браузером чи медіаплеєром, а також не використовується для створення вебконтенту, було вирішено перевірити його на відповідність стандарту WCAG 2.1.

4.1.1. Використане програмне забезпечення

Хоча тестування більшою мірою полягало в ручній перевірці виконання рекомендацій зі стандарту WCAG, для перевірки контрасту основних кольорів компонента та його сумісності з асистивними технологіями було використано наступні інструменти:

- **Contrast Checker** [32]. Онлайн-застосунок, що дозволяє вирахувати співвідношення контрасту між кольорами переднього плану та фону, а також визначає, чи отримане значення відповідає вимогам стандарту WCAG.

  Цей інструмент зручний тим, що він визначає відповідність стандарту WCAG відразу для всіх рівнів відповідності (AA та AAA) і для різних випадків застосування кольорів. Наприклад, для звичайного тексту та жирного тексту, а також для графічних компонентів та елементів інтерфейсу користувача.

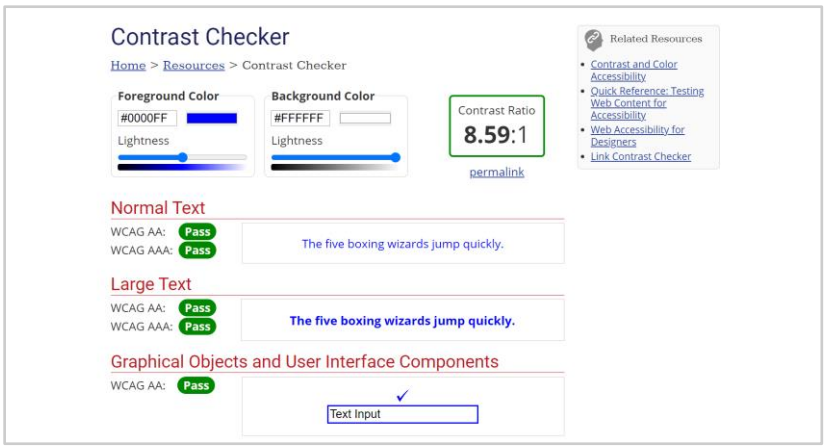


Рисунок 4.2. Інтерфейс інструменту Contrast Checker

- **JAWS Screen Reading Software** [33]. Одна з найбільш популярних програм-читачів екрана для операційної системи Microsoft Windows. JAWS або безпосередньо зачитує текст з екрана, або перетворює його на шрифт Брайля. Під час перевірки компонента Flexmonster був використаний лише функціонал зачитування тексту.

  JAWS належить до платного програмного забезпечення, хоча має і пробну версію. В неї є обмеження на використання, а саме 40 хвилин. Цього часу недостатньо для повноцінної перевірки сумісності свого продукту з асистивними технологіями.
- **Microsoft Windows Narrator** [34]. Програма Windows Narrator теж є читачем екрана. Вона вбудована в операційну систему Windows 10 і добре сумісна з нею. Можна сказати, що Windows Narrator є умовно безкоштовним, адже платити за саму програму не потрібно, лише за операційну систему.

  За бажання, користувач може підлаштувати Windows Narrator під свої потреби, а також використовувати його з оновлюваними дисплеями Брайля.

  Під час перевірки компонента Flexmonster був використаний лише функціонал зачитування тексту.

4.1.2. Виявлені невідповідності стандарту WCAG

Під час тестування компонента Flexmonster на відповідність стандарту WCAG, було виявлено такі невідповідності:

- поганий контраст деяких кольорів фону та переднього плану. Наприклад, в модальному вікні, яке містить список полів, для заголовків та фону використовувалися відтінки сірого, які не дуже відрізнялися між собою, а саме #f7f7f7 для фону та #999 для тексту. Вставивши ці значення у Contrast Checker, отримуємо співвідношення контрасту

2.65:1, в той час як WCAG вимагає як мінімум 4.5:1 для звичайного тексту та 3:1 для жирного [35].

Подібна проблема з контрастом кольорів тексту та фону, фону та граней елементів, заливки іконок та їх контуру спостерігалася в багатьох місцях інтерфейсу користувача. Це означає, що користувачам з вадами зору було б важко читати текст та взаємодіяти з компонентом;

- елементи інтерфейсу, на яких перебуває клавіатурний фокус, ніяк не підсвічувались, коли здійснювалася навігація з клавіатури. Це могло створити незручності у користуванні, адже було незрозуміло, де саме зараз знаходиться користувач;
- коли здійснювалася навігація з клавіатури, клавіатурний фокус переміщався по вкладках на панелі інструментів в неправильному порядку. Правильний порядок вкладок такий: Connect -> Open -> Save -> Export -> Grid -> Charts -> Format -> Options -> Fields -> Fullscreen (рис. 4.3).

Рисунок 4.3. Правильний порядок вкладок на панелі інструментів

Клавіатурний фокус обходив ці елементи в наступному порядку: Format -> Options -> Fields -> Fullscreen -> Connect -> Open -> Save -> Export -> Grid -> Charts (рис. 4.4), що не є очікуваною поведінкою;

Рисунок 4.4. Порядок, в якому фокус обходив вкладки під час тестування

- клавіатурний фокус не потрапляв на деякі елементи інтерфейсу користувача, наприклад, на таблицю. Це означає, що з цими елементами не можна було взаємодіяти з клавіатури;
- під час тестування компонента з використанням JAWS, виникали проблеми з відкриттям деяких вкладок на панелі інструментів з допомогою клавіатури. Не працювали ті вкладки, які містять в собі спадне меню чи відкривають спливні вікна (наприклад, Connect, Export, Charts, тощо). Це пояснюється недостатньою сумісністю компонента з асистивними технологіями;
- коли відкривалося спливне вікно, клавіатурний фокус не переміщався на вікно, а залишався на елементі, що відкрив його (наприклад, кнопка). Таким чином, користувач не міг відразу потрапити у спливне вікно;
- ще однією проблемою зі спливними вікнами було те, що клавіатурний фокус виходив за межі вікна та виділяв елементи, що не належали до нього (наприклад, вкладки панелі інструментів), що створювало незручності для користувача;
- також проблема з клавіатурним фокусом виникала при закритті спливного вікна — він зникав, і потрібно було вручну переміщати фокус туди, де він був перед відкриттям модального вікна. Така поведінка не є передбачуваною, адже після закриття спливного вікна фокус повинен переміщатися на елемент, який відкрив це вікно (наприклад, кнопка);
- роль деяких елементів була неправильно визначена програмами-читачами екранів. Це ставалося у випадках, коли функція використаного HTML-елемента не збігалася з його призначенням. Наприклад, кнопка для видалення правил умовного форматування створена не з використанням тегу <button>, а за допомогою тегу <a>, який відповідає ролі link (посилання). Через це асистивні технології обробляли цю кнопку саме як посилання, що могло заплутати користувача;
- властивості елементів інтерфейсу користувача не були зазначені правильно чи не були зазначені взагалі, через що асистивні технології не

могли правильно обробити ці елементи. Наприклад, читачі екранів не озвучували підписи до деяких елементів інтерфейсу;

- стани елементів інтерфейсу користувача не відслідковувались належним чином для правильної обробки програмами-читачами екранів. Наприклад, не озвучувався поточний стан радіокнопок (обрано кнопку чи ні);
- інструмент Color Picker, який застосовується під час створення правил умовного форматування, також виявився несумісним з асистивними технологіями. Читачі екрана не могли прочитати, які кольори надає інструмент. Хоча навіть якби і змогли, це не мало б значення, адже кольори подані в шістнадцятковому форматі, тож користувач все одно б їх не зрозумів.

4.2. Створення контрастної теми

Щоб покращити контрастність кольорів та зробити компонент Flexmonster зручнішим для людей з вадами зору, було вирішено створити спеціальну доступну CSS-тему, яка б перевизначала стандартні кольори компонента і робила їх більш контрастними. Оскільки створення окремих CSS-тем для компонента є звичним явищем, такий підхід є логічним та знайомим для розробників, що працюють з Flexmonster.

За основу для доступної CSS-теми було обрано тему, яку компонент використовує за замовчуванням. В новій темі було оновлено кольори таких елементів:

- **іконки на панелі інструментів.** Щоб покращити видимість іконок на панелі інструментів, було змінено колір їх контуру з відтінку #999 на відтінок #454545. Колір заливки іконок — #e9e9e9. Співвідношення контрасту між попередніми кольорами заливки та контуру дорівнювало 2.34:1, а між новими — 7.89:1. Це значення відповідає стандарту WCAG на рівні AAA.

Змінився і колір підписів до іконок. Раніше це був відтінок #888, чиє співвідношення контрасту з білим фоном дорівнювало 3.54:1, що лише частково відповідає стандарту WCAG на рівні AA. Новим кольором підписів став відтінок #111, і він контрастує з білим набагато краще — співвідношення контрасту дорівнює 18.88:1, що повністю відповідає стандарту WCAG;

- **розділювачі груп іконок.** Попереднім кольором розділювачів був відтінок #d5d5d5, що контрастував з білим як 1.46:1. Таке значення зовсім не підходило під стандарт, тому новим кольором розділювачів став #111, чиє співвідношення контрасту з білим, як вже зазначалося, дорівнює 18.88:1 і підпадає під стандарт WCAG;

Рисунок 4.5. Приклад оновлених іконок та розділювачів на панелі інструментів

- **іконки на таблиці.** Колір заливки цих іконок було змінено з відтінку #999 на темніший відтінок #454545. Таким чином, зараз значення їхнього контрасту з фоном (відтінок #e9e9e9) дорівнює 7.89:1, що відповідає стандарту WCAG на рівні AAA;

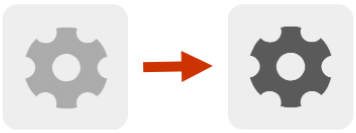


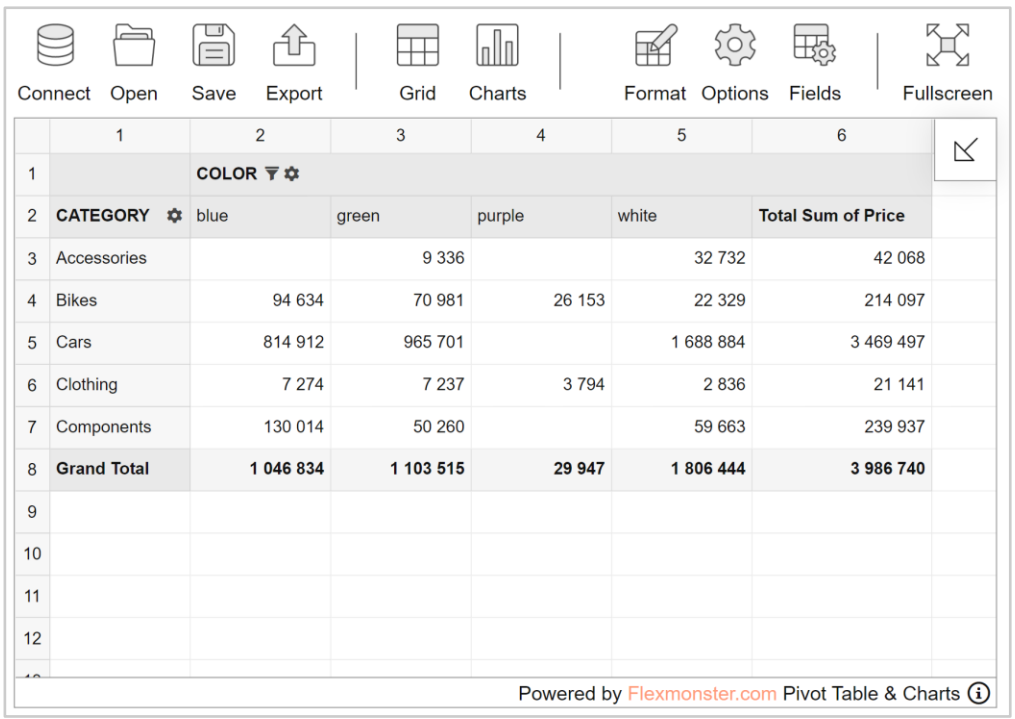
Рисунок 4.6. Приклад оновленої іконки на таблиці

- **допоміжний текст.** Щоб покращити видимість допоміжного тексту (наприклад, текст на панелі брендингу, нумерація рядків та стовпчиків

таблиці), його колір було змінено з відтінку #999 на темніший відтінок #111. Співвідношення контрасту нового відтінку з білим фоном — 18.88:1, з фоном кольору #f7f7f7 — 17.62:1, з фоном кольору #e8e8e8 — 15.41:1. Усі три значення відповідають стандарту WCAG на рівні AAA;

Рисунок 4.7. Приклад оновленого допоміжного тексту

- **границі кнопок, чекбоксів та контейнерів.** Попереднім кольором границь цих елементів був відтінок #d5d5d5, чиє відношення контрасту з білим фоном дорівнювало 1.46:1, а з світло-сірим (відтінок #f7f7f7) — 1.37, що не відповідає стандарту WCAG. Новим кольором границь став відтінок #111, що контрастує з білим як 18.88:1, а з відтінком #f7f7f7 — як 17.62:1, що цілком відповідає стандарту WCAG.



	1	2	3	4	5	6
1		COLOR				
2	CATEGORY	blue	green	purple	white	Total Sum of Price
3	Accessories		9 336		32 732	42 068
4	Bikes	94 634	70 981	26 153	22 329	214 097
5	Cars	814 912	965 701		1 688 884	3 469 497
6	Clothing	7 274	7 237	3 794	2 836	21 141
7	Components	130 014	50 260		59 663	239 937
8	**Grand Total**	**1 046 834**	**1 103 515**	**29 947**	**1 806 444**	**3 986 740**
9						
10						
11						
12						

Рисунок 4.8. Інтерфейс компонента Flexmonster з використанням доступної теми

CSS-тема з підвищеною контрастністю кольорів знаходиться у вільному доступі [36], тож якщо у розробників, які працюють з Flexmonster, виникне

потреба підлаштувати кольори теми під свої вимоги, вони зможуть з легкістю це зробити.

Окрім того, контрастна CSS-тема є цілком сумісною з режимом високої контрастності [37], який наявний в операційній системі Windows. У ньому компонент Flexmonster виглядає так, як показано на рисунку 4.9.

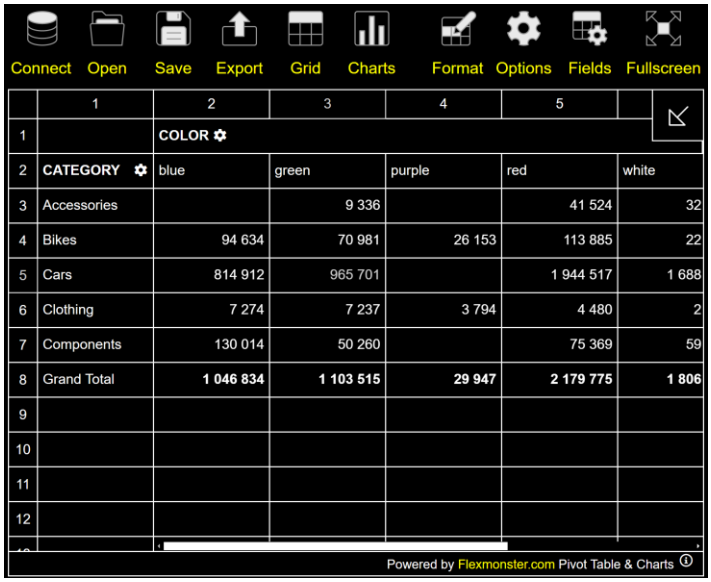


Рисунок 4.9. Зовнішній вигляд компонента Flexmonster у режимі високої контрастності

4.3. Покращення клавіатурної навігації

4.3.1. Підсвітка клавіатурного фокусу

Важливою задачею з поліпшення клавіатурної навігації було підсвічування елементів, на яких знаходиться клавіатурний фокус, адже таким чином користувач міг би розуміти, де він знаходиться.

Проблему відсутності підсвітки клавіатурного фокусу було вирішено з допомогою контрастної CSS-теми. В ній було використано CSS-селектори типу .className:focus [38], які додають чорний контур до елементів, на яких знаходиться клавіатурний фокус:

```
.fm-calculated-view .fm-popup-content .fm-func-btn-group button.fm-calc-action:focus {
    outline: dashed @theme-text-color 2px;
    outline-offset: -2px;
}
```

Таким чином, користувач бачить, на якому елементі перебуває клавіатурний фокус.

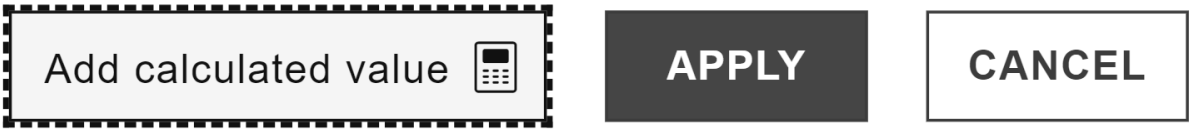


Рисунок 4.10. Приклад елемента, на якому перебуває клавіатурний фокус

4.3.2. Порядок та досяжність елементів під час навігації з клавіатури

За порядок та досяжність елементів під час навігації з клавіатури відповідає HTML-атрибут tabindex [39], що може приймати наступні значення:

- від'ємні значення (найчастіше tabindex="-1"). Від'ємне значення атрибута tabindex для певного елемента означатиме, що цей елемент не можна буде досягти з допомогою клавіатури, а лише з допомогою засобів мови JavaScript;
- нульове значення (tabindex="0"). Таке значення означає, що клавіатурний фокус можна буде перемістити на цей елемент. Порядок елемента визначається його розташуванням в HTML-коді;
- позитивне значення. Позитивний tabindex означає, що клавіатурний фокус можна буде перемістити на цей елемент, а порядок цього елемента визначається значенням атрибута tabindex (чим вище значення, тим раніше елемент отримає клавіатурний фокус). Використовувати позитивні значення для атрибута tabindex не рекомендується, оскільки це порушує логічний порядок навігації: елементи отримуватимуть фокус не в тому порядку, в якому їх бачить користувач. Якщо необхідно змінити порядок елементів, краще зробити це через HTML-код.

За замовчуванням, атрибут tabindex="0" мають такі HTML-елементи, як <a> та <link> (коли мають атрибут href), <button>, <input>, <select>, <textarea> тощо, а, отже, вони є доступними з клавіатури.

Відповідно до результатів тестування компоненту Flexmonster, елементи панелі інструментів отримували клавіатурний фокус в неправильному порядку (рис. 4.11). Втім, вони реалізовані з використанням HTML-тегу <a>, отже, за замовчуванням мають атрибут tabindex="0" і повинні йти в правильному порядку під час навігації з клавіатури.

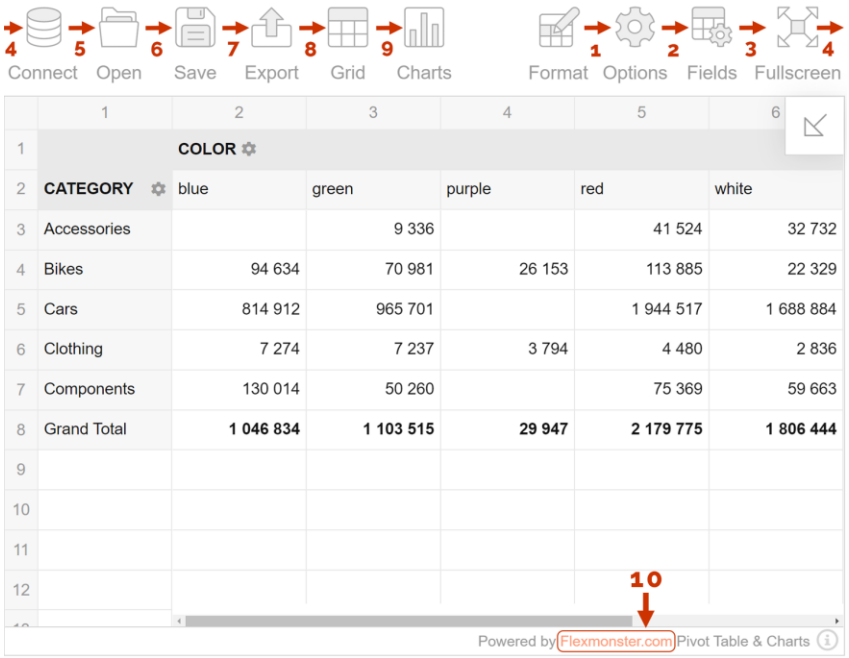


Рисунок 4.11. Порядок, в якому фокус обходив елементи інтерфейсу під час тестування

Як виявилося, проблема з порядком елементів панелі інструментів виникала через те, що в DOM ці елементи були визначені не в тому порядку, в якому розташовувались візуально на сторінці. Отже, ця проблема вирішувалась зміною коду панелі інструментів.

Ще однією проблемою було те, що клавіатурний фокус не потрапляв на деякі елементи інтерфейсу користувача (наприклад, таблицю), через що з ними не можна було взаємодіяти з клавіатури. Це пояснюється тим, що такі елементи інтерфейсу представлені через HTML-елементи, на яких за замовчуванням не може перебувати фокус. Отже, до них було додано атрибут tabindex="0", що і виправило проблему з відсутністю фокусу (рис. 4.12):

```
elem.setAttribute("tabindex", "0");
```

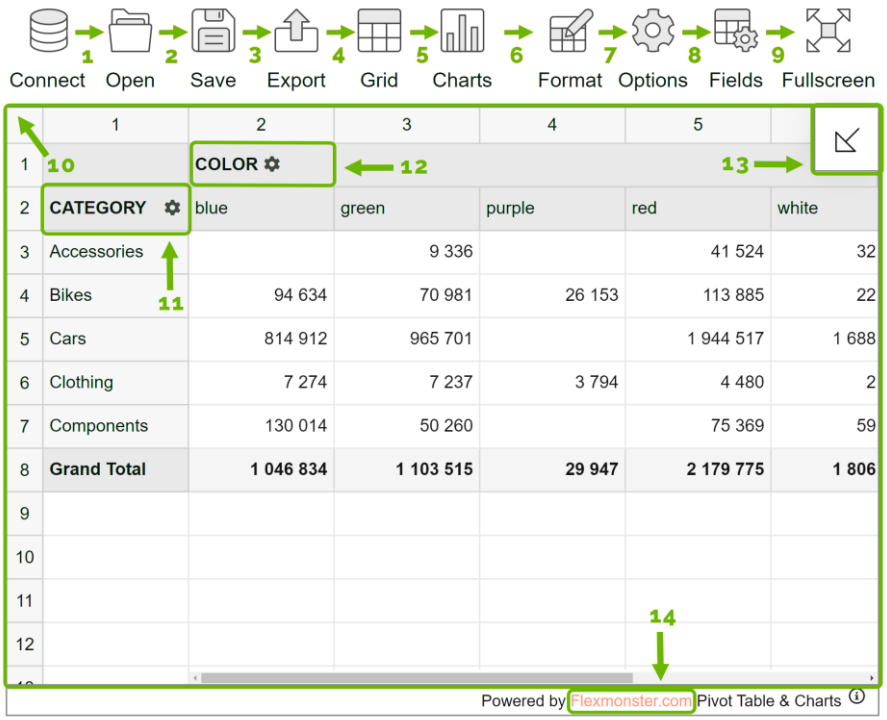


Рисунок 4.12. Новий порядок елементів інтерфейсу

4.3.3. Клавіатурний фокус у спливних вікнах

Під час тестування компонента Flexmonster було виявлено наступні проблеми, що стосувалися спливних вікон:

- коли вікно відкривалося, клавіатурний фокус залишався на елементі, що відкрив вікно (наприклад, кнопка), а мав би бути на самому вікні;
- клавіатурний фокус виходив за межі вікна та виділяв елементи, що не належали до нього (наприклад, вкладки панелі інструментів);
- коли вікно закривалося, клавіатурний фокус зникав, хоча мав би повернутися на елемент, який відкрив це вікно (наприклад, кнопка).

Першу проблему було вирішено попереднім обиранням елемента інтерфейсу, який отримає клавіатурний фокус після відкриття вікна. Так, у конструкторі кожного спливного вікна такий елемент задається через властивість initialFocusAt:

```
dialog.initialFocusAt = textInput;
```

Потім, у момент створення вікна викликається функція initialFocusAt.focus(), яка і переносить клавіатурний фокус на вибраний елемент:

```
if (popup.initialFocusAt) {
    popup.prevActiveElement = document.activeElement;
    popup.initialFocusAt.focus({
        preventScroll: true
    });
}
```

Другу проблему, коли фокус виходив за межі спливного вікна, було вирішено за допомогою двох HTML-елементів , перший з яких знаходиться перед основним контентом вікна, а інший — після. Обидва елементи мають атрибут tabindex="0", отже, на них може перебувати клавіатурний фокус:

```
var focusGuardTop = document.createElement("div");
focusGuardTop.setAttribute("tabindex", "0");
focusGuardTop.addEventListener("focus", keepFocusHandler);
contentPanel.insertBefore(focusGuardTop, contentPanel.firstChild);

var focusGuardBottom = document.createElement("div");
focusGuardBottom.setAttribute("tabindex", "0");
focusGuardBottom.addEventListener("focus", keepFocusHandler);
```

Коли користувач переводить клавіатурний фокус на ці елементи, спрацьовує функція keepFocusHandler() — обробник події focus. Вона за допомогою властивості relatedTarget [40] отримує з об'єкту події попередній елемент, на якому перебував клавіатурний фокус, і переводить фокус на нього:

```
function keepFocusHandler(e) {
    if (e.relatedTarget) {
        e.relatedTarget.focus();
    }
}
```

Таким чином, користувач не може вийти за межі вікна, не закривши його.

Третю проблему, коли клавіатурний фокус зникав при закритті спливного вікна, було вирішено запам'ятовуванням елемента, який відкриває вікно. Це відбувається в момент створення вікна. У властивість об'єкта вікна prevActiveElement з допомогою властивості document.activeElement [41] записується елемент, на якому перебуває клавіатурний фокус, тобто елемент, що відкриває спливне вікно:

```
popup.prevActiveElement = document.activeElement;
```

Коли модальне вікно закривається, викликається функція prevActiveElement.focus(), яка переносить клавіатурний фокус на елемент, що відкрив модальне вікно:

```
if (popup.prevActiveElement) {
    popup.prevActiveElement.focus({ preventScroll: true });
}
```

4.3.4. Інші поліпшення

4.3.4.1. Контекстне меню як альтернативний спосіб доступу до функціоналу

Під час роботи над покращенням клавіатурної навігації виникали випадки, коли потрібно було знайти альтернативний підхід до реалізації наявного в компоненті Flexmonster функціоналу.

Наприклад, якщо навести мишу на клітинку, яка є заголовком рядка чи колонки, з'явиться іконка, натиснувши на яку можна буде посортувати дані за цим рядком чи колонкою. Якщо ж клітинка є назвою поля, то, як показано на рисунку 4.13, з допомогою миші можна відкрити вікно фільтрації або перетягти поле в рядки, колонки, чи взагалі прибрати з таблиці (такий функціонал доступний і у спливному вікні, яке дозволяє керувати усіма полями датасету).

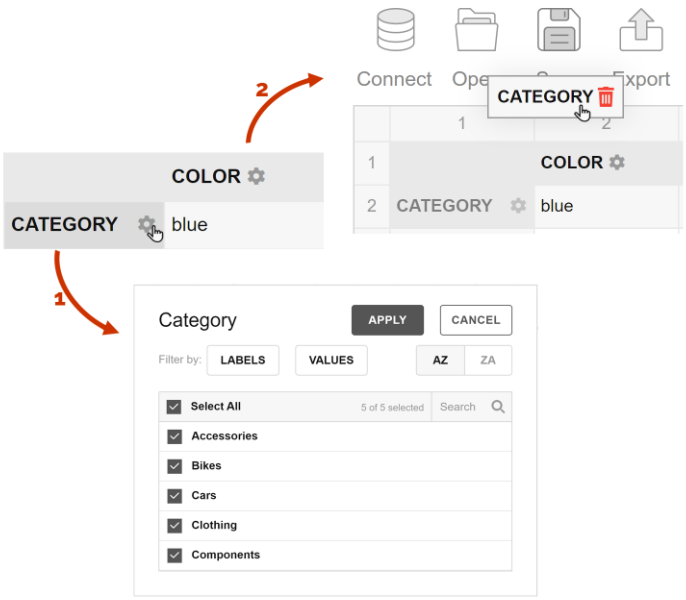


Рисунок 4.13. З допомогою миші можна відкрити вікно фільтрації (1), а також перетягти поле (2)

Реалізація сортування та фільтрації спочатку виглядає простою — потрібно просто додати tabindex="0" до іконок, щоб їх можна було вибрати з допомогою клавіатури, та обробник подій, який сортуватиме дані чи відкриватиме вікно фільтрації залежно від обраної іконки. Втім, такий підхід має наступні недоліки:

1. з однією клітинкою таблиці зазвичай можна робити більше ніж одну дію. Якщо доступ до цих дій організувати через іконки, користувачеві доведеться кілька разів натиснути клавішу Tab, щоб пройти одну клітинку в таблиці. А оскільки таких клітинок може бути досить багато, часу на навігацію по таблиці піде чимало;
2. іконки є досить маленькими, і якщо до них додати той CSS-стиль, який мають інші елементи з клавіатурним фокусом, вони можуть стати нерозбірливими та неестетичними.

Враховуючи ці недоліки, було вирішено надати доступ до функціоналу сортування чи фільтрації через контекстне меню клітинки, що відкривається натисканням клавіші Enter. Такий підхід є логічно виправданим, адже усі клітинки за замовчуванням мають контекстне меню. Таким чином, щоб пройти одну клітинку в таблиці, користувачеві потрібно буде натиснути Tab лише раз, а

якщо він захоче взаємодіяти з клітинкою, він зможе відкрити контекстне меню та обрати потрібну йому опцію (рис. 4.14).

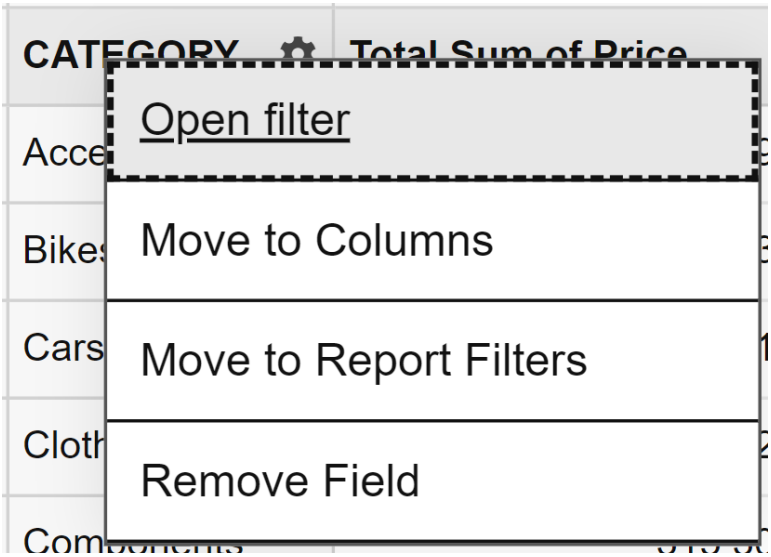


Рисунок 4.14. Вибір опції керування полем через контекстне меню

Аналогічний підхід було вирішено використати і для реалізації функції перетягування полів: користувач відкриває контекстне меню і обирає, куди він хоче перетягнути поле: у рядки, колонки, чи прибрати з таблиці (рис. 4.14).

4.3.4.2. Розширення переліку клавіш, які можна використовувати для навігації по компоненту Flexmonster

Зазвичай для того, щоб дістатися до певного елементу інтерфейсу користувача, використовується клавіша Tab. У випадку з Flexmonster, цієї клавіші може не вистачити для зручного користування компонентом. Щоб зрозуміти це, достатньо розглянути таблицю. Вона може містити у собі сотні клітинок, і тоді користувач може зіткнутися з наступними проблемами:

1. клавіша Tab пересуває клавіатурний фокус у двох напрямках: вліво та вправо. Водночас є логічним, щоб по клітинках таблиці можна було пересуватися у чотирьох напрямках: вгору, вниз, вліво та вправо. Під таку концепцію краще підходять клавіші-стрілки, адже з ними користувач швидше дістанеться до потрібної клітинки;

2. якщо реалізувати навігацію по таблиці лише з допомогою клавіші Tab, то користувачеві доведеться проходити кожну клітинку таблиці, щоб дістатися до елементу, що йде після компоненту Flexmonster.

Щоб уникнути цих незручностей, було вирішено розширити перелік клавіш, з допомогою яких можна взаємодіяти з компонентом (рис. 4.15). Так, клавіша Tab тепер використовується для того, щоб пересуватися основними елементами інтерфейсу користувача — вкладками панелі інструментів, кнопками та контейнерами, — а клавіші-стрілки використовуються для навігації по клітинках таблиці та списках (як звичайних, так і спадних).

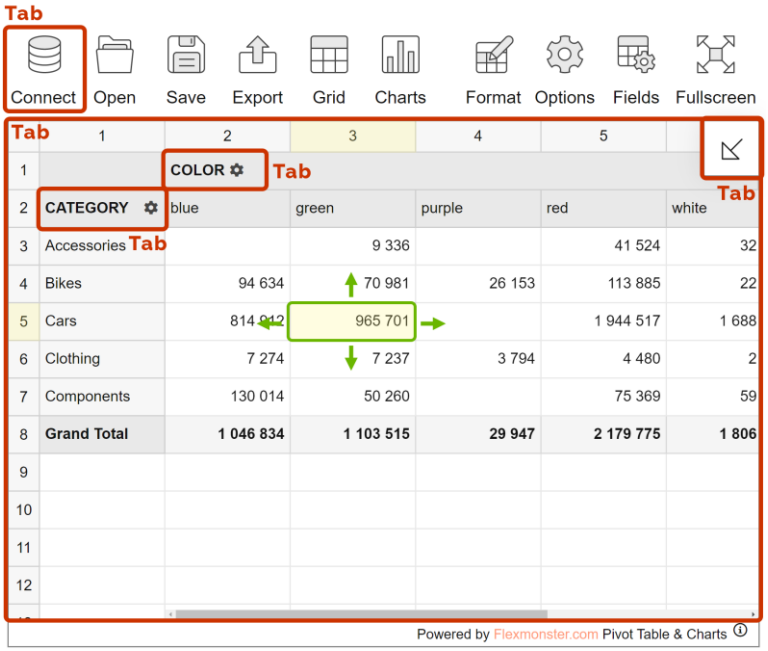


Рисунок 4.15. Розподіл клавіш навігації по компоненту

Таким чином, тепер користувачу не потрібно проходити по всіх клітинках таблиці навіть якщо вони йому не потрібні: достатньо натиснути Tab, щоб перейти з таблиці до наступного елемента інтерфейсу. А якщо користувачеві потрібно взаємодіяти з певною клітинкою, то йому слід обрати таблицю клавішею Tab і натиснути комбінацію клавіш Ctrl + Enter, яка перенесе клавіатурний фокус всередині таблиці.

Код нижче демонструє реалізацію навігації спадним списком з допомогою клавіш-стрілок:

```
var _isOpen = parentTab.classList.contains("fm-opened");
if (_isOpen && (e.keyCode == 38 || e.keyCode == 40)) {
    setActiveItemAt(e.keyCode == 40 ? _activeItemIndex + 1 : _activeItemIndex - 1);
    e.preventDefault();
}
```

Значення змінної _isOpen дорівнює true, якщо елемент списку має CSS-клас "fm-opened" (його наявність означає, що користувач розгорнув список) і false, якщо не має. Якщо список розгорнуто, і користувач натис клавішу "стрілка-вгору" чи "стрілку-вниз" (коди 38 та 40 відповідно [42]), то функція setActiveItemAt() переводить клавіатурний фокус на наступний чи попередній елемент.

Також Flexmonster тепер підтримує такі клавіші: Esc — закриває спливне вікно чи спадний список, пробіл — альтернатива клавіші Enter, та комбінацію клавіш Ctrl + A, яка вибирає всі клітинки таблиці, що не є пустими.

4.4. Покращення сумісності з асистивними технологіями

Щоб покращити сумісність компонента Flexmonster з асистивними технологіями, було вирішено використати фреймворк WAI-ARIA та правильно визначити для елементів інтерфейсу користувача ролі, властивості та стани.

4.4.1. Визначення ролей елементів

Більшість елементів інтерфейсу користувача компоненту Flexmonster реалізовано з використанням HTML-елементів, які не відповідають їх призначенню. Наприклад, клітинки таблиці створені з допомогою тегу , а вкладки панелі інструментів — тегу <a>. Зрозуміло, що якщо явно не прописати ролі для цих елементів, асистивні технології не будуть обробляти їх правильно, що заплутає користувача.

Отже, до всіх елементів інтерфейсу користувача було додано відповідні ролі. Наприклад, grid — для таблиці, gridcell — для клітинок таблиці, row — для

рядків, columnheader та rowheader — для заголовків стовпчиків та рядків, dialog — для спливних вікон, menubar — для панелі інструментів, menuitem — для вкладки на панелі інструментів, button — для кнопки, checkbox — для прапорців і т. д.

Завдяки доданим ролям асистивні технології правильно озвучують елементи. Наприклад, без ролей menubar та menuitem Windows Narrator озвучував вкладки панелі інструментів як посилання, а з цими ролями він озвучує, що вкладка є елементом меню, та називає її порядковий номер (наприклад, друга з десяти). Без ролей, які були додані до таблиці та всіх її елементів, програма-читач екрана просто зачитувала весь вміст таблиці, що звучало дуже незрозуміло. Тепер же асистивні технології озвучують, що це таблиця, і називають кількість рядків та колонок в ній.

4.4.2. Визначення властивостей елементів

Щоб асистивні технології могли коректно зачитувати назви елементів вводу даних та спливних вікон, а також встановлювати зв'язки між певними елементами інтерфейсу користувача, до цих елементів було вирішено додати aria-властивості, які передаватимуть усю інформацію про елемент асистивним технологіям.

Наприклад, було використано такі властивості:

- aria-labelledby та aria-label — ці властивість вказують асистивним технологіям, який підпис має поточний елемент. Такі властивості було додано до спадних списків, щоб користувач зрозумів, що саме вони дозволяють обрати, а також спливних вікон;
- aria-haspopup — цю властивість було додано до елементів, що мають спадне меню чи список (наприклад, вкладки панелі інструментів). Таким чином, асистивні технології озвучуватимуть користувачеві, що поточний елемент сам по собі не виконує якісь дії, а лише відкриває спадне меню;

- aria-controls — аналогічно до попередньої, цю властивість також було додано до елементів зі спадними списками та меню. Завдяки ній асистивні технології можуть зрозуміти, появу якого меню чи списку контролює поточний елемент;
- aria-activedescendant — цю властивість також було додано до елементів, що мають спадне меню чи список. Вона дає асистивним технологіям зрозуміти, який саме елемент повинен отримати фокус при відкритті меню чи списку, та правильно обробити його фокусування.

4.4.3. Визначення станів елементів

Щоб асистивні технології могли коректно інформувати користувача про поточний стан радіокнопок, прапорців, спадних списків та інших елементів, до цих елементів додано aria-властивості, які передаватимуть усю інформацію про стан елемента асистивним технологіям.

Наприклад, було використано такі стани:

- aria-expanded — дана властивість показує, чи розгорнуто спадний список;
- aria-disabled — ця властивість інформує асистивні технології, чи є поточний елемент активним, чи ні. Наприклад, якщо користувач хоче під'єднатися до OLAP-кубу, він не зможе обрати кнопку для з'єднання, поки не заповнить усі потрібні поля;
- aria-checked — дана властивість інформує асистивні технології, чи вибрано даний прапорець;
- aria-hidden="true" — ця властивість інформує API браузера про те, що не потрібно сповіщати асистивні технології про про наявність поточного елементу. Наприклад, цю властивість використано, щоб "приховати" від асистивних технологій іконки з панелі інструментів.

Завдяки станам aria-expanded та aria-checked програма Windows Narrator правильно озвучує стани елементів. Так, після прочитання назви списку чи

прапорця, програма озвучує його роль (наприклад, checkbox) та стан (наприклад, checked).

4.4.4. Робота з інструментом Color Picker

Під час тестування компонента виявилося, що Інструмент Color Picker не є зручним для людей, що користуються асистивними технологіями, адже усі наявні в ньому кольори були подані в шістнадцятковому форматі.

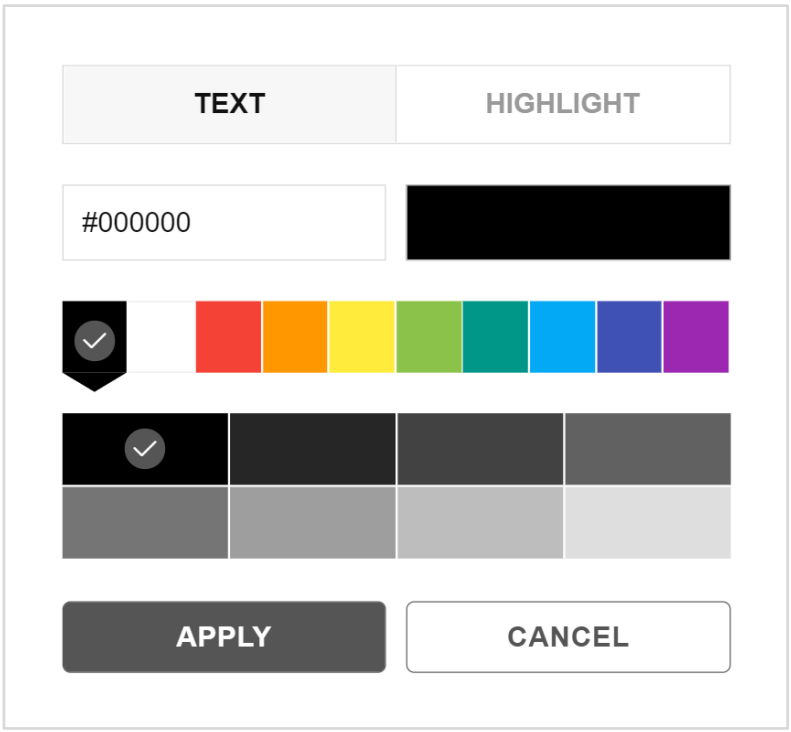


Рисунок 4.16. Інструмент Color Picker

Просто присвоїти кожному кольору якусь назву не вийшло б, адже кожен основний колір має вісім відтінків, і придумати для кожного з них зрозумілу назву нереально.

Втім, задача інструмента Color Picker не в тому, щоб точно передавати відтінки кольорів, а в тому, щоб давати можливість обрати з якомога більшого набору кольорів. Отже, було вирішено дати зрозумілі назви основним кольорам (наприклад, black чи orange), а їхні відтінки просто пронумерувати, щоб дати користувачу зрозуміти, що це — різні відтінки одного кольору. Таким чином,

кожен відтінок має назву типу <основний_колір> Shade <номер_відтінку> (наприклад, black Shade 1).

Якщо ж вдатися до роздумів над концепцією умовного форматування, для якого і використовується Color Picker, то можна зрозуміти, що основна його ціль — не розфарбувати клітинки таблиці в різні кольори, а передати через ці кольори певну інформацію. Наприклад, якщо з допомогою таблиці аналізується рівень забрудненості повітря в Києві, червона клітинка означала б, що повітря було дуже забруднене, а зелена — що забрудненість була в межах норми.

В такому випадку, клітинки з підписами кольорів типу red Shade 1 не дали б сліпій людині, яка користується асистивними технологіями, повного розуміння поданої інформації, адже вона може просто не мати з червоним кольором тих асоціацій, які є в зрячих людей.

З цієї точки зору, ліпше було б присвоювати клітинці не колір, а якесь додаткове повідомлення. Наприклад, "небезпека", "стан задовільний", чи інше повідомлення, яке відповідатиме контексту.

Ідея такої подачі інструменту Color Picker поки знаходиться на стадії розробки, тож це вдосконалення буде реалізовано в найближчому майбутньому.

4.5. Висновки

У даному розділі описано процес адаптації JavaScript-компоненту Flexmonster Pivot Table & Charts до потреб людей з обмеженими можливостями. Для цього було проведено ручне тестування компоненту на відповідність стандарту WCAG 2.1.

Серед недоліків були: недостатній контраст кольорів деяких елементів інтерфейсу користувача, неможливість доступитися до всього функціоналу компоненту з допомогою клавіатури, а також погана сумісність з асистивними технологіями.

Щоб виправити недоліки, виявлені в ході тестування Flexmonster, було створено контрастну тему, поліпшено клавіатурну навігацію, та використано фреймворк WAI-ARIA для забезпечення сумісності з асистивними технологіями.

ВИСНОВКИ

У ході виконання даної роботи було виконано наступні завдання:

- проаналізовано сучасний стан вебсайтів у контексті вебдоступності;
- проаналізовано, які існують вади здоров'я та які труднощі у користуванні мережею Інтернет вони викликають;
- досліджено існуючі стандарти та найкращі практики у сфері вебдоступності;
- проведено тестування JavaScript-компонента Flexmonster Pivot Table & Charts на відповідність стандарту вебдоступності WCAG 2.1;
- проведено роботу над виправленням знайдених невідповідностей з використанням найкращих практик вебдоступності, в тому числі фреймворку WAI-ARIA.

Досягнуто наступні цілі:

- JavaScript-компонент Flexmonster Pivot Table & Charts відповідає стандартам вебдоступності, а також враховує усі потреби користувачів з обмеженими можливостями.

Очікується, що відповідність компонента Flexmonster Pivot Table & Charts стандартам вебдоступності збільшить коло людей, які зможуть користуватися цим продуктом, та сприятиме створенню позитивного образу компанії, яка турбується про своїх клієнтів. Як наслідок, це зміцнить довіру людей до компанії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Digital around the world – DatePortal. URL: Https://Datareportal.Com/Global-Digital-Overview (дата звернення: 10.04.2021).
2. Disability and health – World Health Organization. URL: https://www.who.int/news-room/fact-sheets/detail/disability-and-health (дата звернення: 10.04.2021).
3. Соціальний захист населення у 2019 році – Державна служба статистики України. URL: http://ukrstat.gov.ua/druk/publicat/kat_u/2020/zb/07/zb_szn_2019.pdf (дата звернення: 10.04.2021).
4. Introduction to Web Accessibility – W3C. URL: https://www.w3.org/WAI/fundamentals/accessibility-intro/ (дата звернення: 10.04.2021).
5. Nucleus Research Note: The Internet is Unavailable – Deque. URL: https://accessibility.deque.com/nucleus-accessibility-research-2019 (дата звернення: 10.04.2021).
6. The Internet Is for Everyone, Right? Not With a Screen Reader – Wired. URL: https://www.wired.com/story/web-accessibility-blind-users-dominos/ (дата звернення: 10.04.2021).
7. The WebAIM Million – WebAIM. URL: https://webaim.org/projects/million/ (дата звернення: 10.04.2021).
8. The Click-Away Pound Report 2019 – Click Away Pound. URL: http://www.clickawaypound.com/downloads/cap19final0502.pdf (дата звернення: 10.04.2021).
9. How People with Disabilities Use the Web – W3C. URL: https://www.w3.org/WAI/EO/Drafts/WAI-access-profiles (дата звернення: 10.04.2021).

10. Blindness and visual impairment – World Health Organization. URL: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (дата звернення: 10.04.2021).
11. World report on hearing – World Health Organization. URL: https://cdn.who.int/media/docs/default-source/documents/health-topics/deafness-and-hearing-loss/world-report-on-hearing/wrh-executive-summary.en.pdf (дата звернення: 10.04.2021).
12. Web Content Accessibility Guidelines (WCAG) Overview – W3C. URL: https://www.w3.org/WAI/standards-guidelines/wcag/ (дата звернення: 10.04.2021).
13. W3C – W3C. URL: https://www.w3.org/ (дата звернення: 10.04.2021).
14. Web Accessibility Initiative – W3C. URL: https://www.w3.org/WAI/ (дата звернення: 10.04.2021).
15. Web Content Accessibility Guidelines (WCAG) 2.0 – W3C. URL: https://www.w3.org/TR/WCAG20/ (дата звернення: 10.04.2021).
16. Web Content Accessibility Guidelines (WCAG) 2.1 – W3C. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 10.04.2021).
17. Authoring Tool Accessibility Guidelines (ATAG) Overview – W3C. URL: https://www.w3.org/WAI/standards-guidelines/atag/ (дата звернення: 10.04.2021).
18. Authoring tool – W3C. URL: https://www.w3.org/TR/ATAG20/#def-Authoring-Tool (дата звернення: 10.04.2021).
19. User Agents Accessibility Guidelines (UAAG) Overview – W3C. URL: https://www.w3.org/WAI/standards-guidelines/uaag/ (дата звернення: 10.04.2021).
20. WAI-ARIA Overview – W3C. URL: https://www.w3.org/WAI/standards-guidelines/aria/ (дата звернення: 10.04.2021).
21. Information and Communication Technology – U. S. Access Board. URL: https://www.access-board.gov/ict.html (дата звернення: 10.04.2021).
22. Quick Reference Guide to Section 508 Checklist – Deloitte. URL: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/public-sector/us-

fedinnov-quick-reference-guide-to-section-508-checklist.pdf (дата звернення: 10.04.2021).

23. Side by Side WCAG vs. 508 – JimThatcher. URL: https://jimthatcher.com/sidebyside.htm (дата звернення: 10.04.2021).
24. Web Accessibility Laws in the United States in 2020 – Rev. URL: https://www.rev.com/blog/web-accessibility-laws-in-the-u-s (дата звернення: 10.04.2021).
25. Standard on Web Accessibility – Government of Canada. URL: https://www.tbs-sct.gc.ca/pol/doc-eng.aspx?id=23601 (дата звернення: 10.04.2021).
26. Accessibility requirements for ICT products and services – ETSI. URL: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.01.01_60/en_301549v030101p.pdf (дата звернення: 10.04.2021).
27. ISO 30071-1 digital accessibility standard – Hassell Inclusion. URL: https://www.hassellinclusion.com/iso-30071-1/ (дата звернення: 10.04.2021).
28. Постанова "Про внесення змін до деяких постанов Кабінету Міністрів України щодо функціонування офіційних веб-сайтів органів виконавчої влади" від 12 червня 2019 р. № 493 – Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/493-2019-%D0%BF#Text (дата звернення: 10.04.2021).
29. Flexmonster Pivot Table & Charts – Flexmonster. URL: https://www.flexmonster.com/ (дата звернення: 10.04.2021).
30. Accessible Rich Internet Applications (WAI-ARIA) 1.1 – W3C. URL: https://www.w3.org/TR/wai-aria-1.1/ (дата звернення: 10.04.2021).
31. When should you use WAI-ARIA? – MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics#when_should_you_use_wai-aria (дата звернення: 10.04.2021).
32. Contrast Checker – WebAIM. URL: https://webaim.org/resources/contrastchecker/ (дата звернення: 10.04.2021).

33. JAWS – Freedom Scientific. URL: https://www.freedomscientific.com/products/software/jaws/ (дата звернення: 10.04.2021).
34. Complete guide to Narrator – Microsoft | Support. URL: https://support.microsoft.com/en-us/windows/complete-guide-to-narrator-e4397a0d-ef4f-b386-d8ae-c172f109bdb1 (дата звернення: 10.04.2021).
35. Success Criterion 1.4.3 Contrast (Minimum) – W3C. URL: https://www.w3.org/TR/WCAG21/#contrast-minimum (дата звернення: 10.04.2021).
36. Accessible CSS theme for Flexmonster – Flexmonster. URL: https://cdn.flexmonster.com/theme/accessible/flexmonster.css (дата звернення: 10.04.2021).
37. Use high contrast mode in Windows 10 – Microsoft | Support. URL: https://support.microsoft.com/en-us/windows/use-high-contrast-mode-in-windows-10-fedc744c-90ac-69df-aed5-c8a90125e696 (дата звернення: 10.04.2021).
38. CSS :focus Selector – W3SCHOOLS. URL: https://www.w3schools.com/cssref/sel_focus.asp (дата звернення: 10.04.2021).
39. tabindex – MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/tabindex (дата звернення: 10.04.2021).
40. relatedTarget – MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent/relatedTarget (дата звернення: 10.04.2021).
41. activeElement – MDN Web Docs. URL: https://developer.mozilla.org/ru/docs/Web/API/Document/activeElement (дата звернення: 10.04.2021).
42. Key codes – Keycode. URL: https://keycode.info/ (дата звернення: 10.04.2021).