МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «УНІВЕРСИТЕТ «КРОК»

Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА

за темою

«Розробка та створення сайту для надання послуг у сфері туризму»

Студент 4 курсу групи КН-20К

Савчук Дмитро Тарасович

(прізвище, ім'я та по-батькові студента)

Керівник дипломної роботи

Викладач

(посада керівника)

Кириченко Віктор Вікторович

(прізвище, ім'я та по-батькові керівника)

До захисту

(резолюція «До захисту»)

(підпис студента)

11.06.2024

(дата)

(підпис викладача)

Київ, 2024 рік

Зміст

Вступ

У сучасному світі туризм відіграє важливу роль в економічному розвитку багатьох країн, а також у формуванні культурного обміну між народами. З розвитком інформаційних технологій і всесвітньої мережі Інтернет, більшість туристичних послуг перейшли в онлайн-простір. Веб-сайти туристичних агентств, онлайн-платформи для бронювання готелів і квитків, а також спеціалізовані туристичні блоги стали невід'ємною частиною індустрії.

Створення сайту для надання послуг у сфері туризму є важливим кроком для будь-якого туристичного бізнесу, оскільки дозволяє забезпечити зручний доступ до інформації, підвищити рівень обслуговування клієнтів і розширити аудиторію. Основною метою такого сайту є надання користувачам повної та актуальної інформації про туристичні послуги, маршрути, готелі, визначні місця та інші аспекти подорожей.

Однак розробка ефективного туристичного сайту вимагає не лише технічних знань у сфері веб-розробки, але й розуміння специфіки туристичного бізнесу. Одним з ключових аспектів є управління веб-контентом, яке включає створення, редагування, публікацію та організацію інформації на сайті. Якісний веб-контент не лише приваблює потенційних клієнтів, але й забезпечує їм корисну та актуальну інформацію, що сприяє формуванню позитивного іміджу компанії.

У даній дипломній роботі буде розглянуто процес створення сайту для надання послуг у сфері туризму, зокрема аспекти управління веб-контентом. Основна увага приділятиметься розробці структури сайту, вибору технологій для реалізації, методам оптимізації контенту для пошукових систем та забезпеченню зручності використання для кінцевих користувачів.

Метою цієї роботи є створення ефективного інструменту для надання туристичних послуг, який буде відповідати сучасним вимогам ринку та забезпечить високий рівень задоволеності клієнтів.

Розділ 1: Огляд технологій

У цьому розділі розглянемо сучасні технології, які використовуються для створення та управління веб-сайтами, зокрема для надання послуг у сфері туризму. Це допоможе визначити найкращі інструменти для реалізації проекту та зрозуміти, як кожна з них може сприяти досягненню поставлених цілей.

1.1 Next.js

Next.js – це популярний фреймворк для React, який дозволяє створювати серверні та статичні веб-додатки. Основні можливості Next.js включають:

- Рендеринг на сервері (SSR): Дозволяє рендерити сторінки на сервері, що покращує SEO та час завантаження сторінок.
- Статична генерація (SSG): Можливість попередньо генерувати сторінки під час збірки, що забезпечує швидке завантаження.
- API роутингу: Спрощує створення API для вашого додатка прямо у фреймворку.
- Підтримка TypeScript: Вбудована підтримка TypeScript дозволяє писати надійний та масштабований код.

Next.js є потужним інструментом для створення високопродуктивних, SEO-дружніх веб-додатків.

1.2 Radix UI

Radix UI – це набір низькорівневих компонентів для React, які забезпечують високий рівень доступності та настроюваності. Основні можливості Radix UI включають:

• Аксесибельність: Всі компоненти створені з урахуванням кращих практик доступності, що забезпечує зручність використання для всіх користувачів.

• Настроюваність: Компоненти можна легко налаштовувати відповідно до дизайну вашого проекту.

• Модульність: Кожен компонент є незалежним модулем, що дозволяє використовувати тільки необхідні частини.

Radix UI є відмінним вибором для створення доступних і настроюваних інтерфейсів користувача.

1.3 Tailwind CSS

Tailwind CSS – це утилітарний CSS-фреймворк, який дозволяє швидко створювати індивідуальні дизайни без написання кастомного CSS. Основні можливості Tailwind CSS включають:

• Утилітарний підхід: Використання класів, які безпосередньо застосовуються до елементів для створення стилів.

• Адаптивний дизайн: Вбудована підтримка адаптивного дизайну з використанням брейкпоінтів.

• Настроюваність: Легко настроюється відповідно до потреб проекту через конфігураційний файл.

Tailwind CSS значно прискорює процес розробки та дозволяє створювати узгоджені та гнучкі дизайни.

1.4 Zustand

Zustand – це легковаговий та гнучкий стейт-менеджер для React, який дозволяє ефективно керувати станом додатка. Основні можливості Zustand включають:

• Простота використання: Інтуїтивний API для створення та управління станом.

• Маленький розмір: Zustannd є дуже легким, що робить його швидким і ефективним.

• Сумісність з React: Легко інтегрується з React-компонентами та іншими стейт-менеджерами.

Zustand є відмінним вибором для управління станом у невеликих та середніх проектах, забезпечуючи високу продуктивність та простоту використання.

1.5 shadcn/ui

shadcn/ui – це сучасний набір компонентів для React, який забезпечує високу якість дизайну та функціональність. Основні можливості shadcn/ui включають:

• Якісний дизайн: Компоненти створені з урахуванням сучасних трендів дизайну, що забезпечує привабливий вигляд додатків.

• Висока функціональність: Великий набір готових до використання компонентів, які можна легко інтегрувати в проект.

• Настроюваність: Можливість налаштовувати компоненти під індивідуальні потреби проекту.

shadcn/ui є корисним інструментом для швидкого створення красивих та функціональних інтерфейсів користувача.

Висновок

Огляд цих технологій показує, що кожна з них відіграє важливу роль у створенні ефективного та привабливого веб-сайту для надання послуг у сфері туризму. Використання Next.js, Radix UI, Tailwind CSS, Zustand та shadcn/ui дозволяє створювати високопродуктивні, адаптивні та зручні у використанні веб-додатки, що відповідають сучасним вимогам та потребам користувачів.

Розділ 2: Планування проекту

У цьому розділі буде викладено ключові аспекти планування проекту створення веб-сайту для надання послуг у сфері туризму. Це включає визначення мети та задач проекту, ідентифікацію цільової аудиторії та формулювання функціональних вимог.

2.1 Мета і задачі

Мета проекту: Основною метою проекту є створення веб-сайту, який забезпечить зручний та інформативний доступ до туристичних послуг, сприятиме збільшенню кількості клієнтів і підвищенню їх задоволеності, а також допоможе розширити ринок збуту туристичних продуктів.

Задачі проекту:

1. Розробка структури сайту: Визначення логічної структури сайту, яка включатиме основні розділи та підрозділи.
2. Дизайн інтерфейсу: Створення привабливого та інтуїтивно зрозумілого дизайну, який забезпечить зручність користування.
3. Розробка функціональності: Реалізація необхідних функціональних компонентів, таких як пошук турів, бронювання послуг, відгуки клієнтів тощо.
4. Інтеграція з системами бронювання: Підключення до зовнішніх систем бронювання та платіжних шлюзів для забезпечення можливості онлайн-замовлення.
5. Оптимізація для пошукових систем (SEO): Впровадження SEO-стратегій для покращення видимості сайту в пошукових системах.
6. Тестування та випробування: Проведення всебічного тестування для забезпечення стабільної та безпомилкової роботи сайту.
7. Запуск та підтримка: Запуск сайту в експлуатацію та забезпечення його постійної підтримки та оновлення.

2.2 Цільова аудиторія

Цільова аудиторія проекту включає:

1. Туристи: Люди, які планують подорожі та шукають інформацію про тури, готелі, визначні місця тощо. Вони можуть бути як індивідуальними мандрівниками, так і сімейними туристами чи організованими групами.

2. Бізнес-мандрівники: Особи, які подорожують з діловими цілями та шукають відповідні послуги, такі як бронювання готелів, транспортних засобів і місць проведення заходів.

3. Туристичні агенції та туроператори: Компанії, які займаються організацією турів та подорожей, можуть використовувати сайт для пошуку партнерів і просування своїх послуг.

4. Місцеві жителі: Люди, які шукають інформацію про локальні туристичні послуги та розваги, включаючи екскурсії, ресторани та культурні заходи.

2.3 Функціональні вимоги

Основні функціональні вимоги до сайту включають:

1. Пошук турів:

- Можливість пошуку турів за різними критеріями (країна, місто, тип подорожі, ціна, дати тощо).
- Фільтрація та сортування результатів пошуку.

2. Інформація про тури:

- Детальний опис турів, включаючи маршрут, програму, умови проживання та харчування, ціни та додаткові послуги.
- Відгуки та рейтинги від попередніх клієнтів.

3. Бронювання послуг:

- Онлайн-бронювання турів та інших туристичних послуг (готелі, транспорт, екскурсії).

- Інтеграція з платіжними системами для проведення онлайн-оплат.

4. Користувацький профіль:

- Реєстрація та авторизація користувачів.
- Управління особистими даними, історією бронювань, збереженими турами та відгуками.

5. Інформаційний контент:

- Блог з корисними статтями про подорожі, поради туристам, огляди місць та маршрутів.
- Новини та акції компанії.

6. Мультимедійний контент:

- Галереї з фотографіями та відео з місць призначення та турів.
- Інтерактивні карти з позначками місць призначення.

7. Підтримка клієнтів:

- Онлайн-чат для консультацій.
- Система запитів та відповідей (FAQ).

8. Мультимовність:

- Підтримка кількох мов для обслуговування міжнародних клієнтів.

9. Оптимізація для пошукових систем (SEO):

- Використання ключових слів та мета-тегів.
- Дружні URL-адреси та адаптивний дизайн для мобільних пристроїв.

Висновок

Планування проекту є важливим етапом, який забезпечує чітке розуміння мети, задач, цільової аудиторії та функціональних вимог до веб-сайту. Цей розділ закладає основу для подальших етапів розробки, тестування та впровадження проекту, спрямованого на створення ефективного інструменту для надання туристичних послуг.

?

Розділ 3: Дизайн сайту

У цьому розділі розглянемо основні етапи дизайну сайту для надання послуг у сфері туризму, включаючи процес прототипування, вибір кольорової гами та шрифтів, а також створення інтерфейсу користувача. Це допоможе забезпечити привабливість, зручність та ефективність веб-сайту.

3.1 Прототипування

Прототипування є ключовим етапом у процесі розробки дизайну сайту, який передбачає створення візуальних моделей сторінок сайту. Основні етапи прототипування включають:

1. Створення вайрфреймів:

- Вайрфрейми (wireframes) – це схематичні зображення сторінок сайту, які показують розташування основних елементів, таких як заголовки, текстові блоки, кнопки, зображення та форми.
- Вайрфрейми дозволяють швидко оцінити структуру сторінок і внести необхідні зміни перед початком розробки детального дизайну.

2. Високоточні прототипи:

- Високоточні прототипи (high-fidelity prototypes) є більш деталізованими версіями вайрфреймів, які включають кольорові схеми, шрифти та інтерактивні елементи.
- Ці прототипи допомагають більш точно уявити, як буде виглядати і працювати кінцевий продукт, і є корисними для тестування користувацького досвіду.

3. Інтерактивні прототипи:

- Інтерактивні прототипи дозволяють користувачам взаємодіяти з дизайном, імітуючи функціональність сайту.
- Це допомагає отримати зворотній зв'язок від користувачів на ранніх етапах розробки та зробити необхідні коригування.

3.2 Вибір кольорової гами і шрифтів

Вибір кольорової гами та шрифтів є важливим аспектом дизайну, який впливає на візуальне сприйняття сайту та користувацький досвід.

Кольорова гама:

1. Основні кольори:

- Вибір основних кольорів, які будуть використовуватись у всьому сайті, включаючи фон, текст та основні елементи інтерфейсу.
- Для туристичного сайту часто використовуються теплі та приємні кольори, які асоціюються з подорожами, природою та відпочинком (наприклад, блакитний, зелений, оранжевий).

2. Допоміжні кольори:

- Вибір додаткових кольорів, які використовуються для виділення важливих елементів (кнопок, посилань) та створення контрасту.
- Допоміжні кольори повинні гармонійно поєднуватись з основними кольорами і підкреслювати важливі деталі без перевантаження дизайну.

Шрифти:

1. Основний шрифт:

- Вибір основного шрифту для заголовків та основного тексту. Важливо, щоб шрифт був читабельним і відповідав стилю сайту.
- Для туристичних сайтів часто використовуються сучасні та елегантні шрифти, такі як Sans-serif для заголовків та Serif для основного тексту.

2. Додаткові шрифти:

- Використання додаткових шрифтів для акцентів або спеціальних елементів, таких як цитати або підписи.
- Важливо, щоб додаткові шрифти гармоніювали з основними шрифтами і не створювали візуального шуму.

3.3 Дизайн інтерфейсу користувача

Дизайн інтерфейсу користувача (UI) – це процес створення візуального компонента сайту, що включає розташування елементів, вибір кольорів, шрифтів та стилів.

1. Головна сторінка:

- Заголовок: Великий і яскравий заголовок з ключовою інформацією про компанію та її послуги.
- Навігація: Інтуїтивно зрозуміла навігаційна панель, яка дозволяє користувачам легко переміщатися між розділами сайту.
- Основний контент: Використання великих зображень, відео та текстових блоків для представлення основних послуг та акцій.

2. Сторінка турів:

- Фільтри та сортування: Інтерактивні фільтри для пошуку турів за різними критеріями.
- Картки турів: Візуально привабливі картки з короткою інформацією про тури, включаючи зображення, назву, ціну та короткий опис.

3. Сторінка бронювання:

- Форми: Прості та зрозумілі форми для введення даних користувачів (дата, кількість осіб, особливі побажання).
- Кнопки дій: Яскраві та помітні кнопки для підтвердження бронювання та оплати.

4. Користувацький профіль:

- Особисті дані: Розділ для перегляду та редагування особистих даних.
- Історія бронювань: Список всіх попередніх бронювань з можливістю перегляду деталей та повторного замовлення.

5. Додаткові елементи:

- Чат підтримки: Інтерактивний чат для швидкої консультації з представниками компанії.
- Блок відгуків: Розділ з відгуками клієнтів для підвищення довіри до компанії.

Висновок

Дизайн сайту є критично важливим етапом у створенні ефективного туристичного веб-ресурсу. Прототипування допомагає визначити структуру та функціональність сайту на ранніх етапах розробки. Вибір кольорової гами та шрифтів впливає на візуальне сприйняття та загальне враження від сайту. Дизайн інтерфейсу користувача забезпечує зручність та інтуїтивність взаємодії користувачів із сайтом. Разом ці елементи створюють основу для успішного веб-сайту, що задовольняє потреби користувачів і бізнесу.

Розділ 4: Розробка сайту

У цьому розділі буде детально розглянуто етапи розробки сайту для надання послуг у сфері туризму, зокрема використання технологій Next.js, Tailwind CSS, Zustand та UI бібліотеки shadcn. Це дозволить зрозуміти, як ці інструменти допомагають у створенні сучасного, ефективного та привабливого веб-сайту.

4.1 Структура Next.js

Next.js – це фреймворк для React, який спрощує створення серверних та статичних веб-додатків. Основні компоненти структури Next.js включають:

1. Структура папок:

- pages: Містить всі сторінки додатка. Файли в цій папці автоматично стають маршрутами сайту.
- public: Містить статичні файли, такі як зображення, шрифти та інші ресурси.
- components: Містить багаторазові компоненти, які можна використовувати на різних сторінках.
- styles: Містить файли зі стилями. Хоча ми використовуємо Tailwind CSS, тут можуть бути додаткові стилі.
- api: Містить API маршрути, які дозволяють створювати серверні функції без необхідності створення окремого сервера.

2. Файли конфігурації:

- next.config.js: Файл конфігурації Next.js для налаштування різних параметрів проекту.
- tailwind.config.js: Файл конфігурації Tailwind CSS для налаштування стилів.

3. Особливості:

- Серверний рендеринг (SSR): Дозволяє рендерити сторінки на сервері для покращення SEO та продуктивності.

- Статична генерація (SSG): Дозволяє попередньо генерувати сторінки під час збірки для швидшого завантаження.

4.2 Стилізація за допомогою Tailwind CSS

Tailwind CSS – це утилітарний CSS-фреймворк, який дозволяє створювати індивідуальні дизайни без написання кастомного CSS. Основні кроки для використання Tailwind CSS включають:

1. Інсталяція:

- Встановлення Tailwind CSS через npm або yarn:

bash

```
npm install tailwindcss
npx tailwindcss init
```

2. Налаштування:

- Додавання Tailwind до вашого CSS-файлу:

css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

3. Конфігурація:

- Налаштування файлу tailwind.config.js для додавання кастомних кольорів, шрифтів та інших налаштувань.

4. Використання класів:

- Додавання утилітарних класів безпосередньо до HTML-елементів:

jsx

```
<div className="bg-blue-500 text-white p-4 rounded">
  Привіт, Tailwind CSS!
</div>
```

4.3 Інтерактивність за допомогою Zustand

Zustand – це легковаговий стейт-менеджер для React, який дозволяє ефективно керувати станом додатка. Основні кроки для використання Zustand включають:

1. Інсталяція:

- Встановлення Zustand через npm або yarn:

bash

```
npm install zustand
```

2. Створення сховища:

- Створення файлу сховища для управління станом:

jsx

```
import create from 'zustand';

const useStore = create(set => ({
  тури: [],
  додатиТур: (тур) => set(state => ({ тури: [...state.тури, тур] }))
}));
```

3. Використання сховища:

- Використання стану у ваших компонентах:

jsx

```
import useStore from './store';

const ТурСписок = () => {
  const тури = useStore(state => state.тури);
  return (
    <div>
      {тури.map((тур, індекс) => (
```

```
        <div key={індекс}>{тур.name}</div>
      ))}
    </div>
  );
};
```

4.4 Використання UI Shadcn

Shadcn – це сучасна бібліотека UI компонентів для React, яка забезпечує високу якість дизайну та функціональність. Основні кроки для використання shadcn включають:

1. Інсталяція:

- Встановлення бібліотеки shadcn через npm або yarn:

bash

npm install @shadcn/ui

Імпорт компонентів:

- Імпорт необхідних компонентів у ваш проект:

jsx

```
import { Button, Card, Modal } from '@shadcn/ui';

const МійКомпонент = () => {
  return (
    <div>
      <Card>
        <h2>Туристичний Карта</h2>
        <p>Опис туру...</p>
```

```
        <Button>Забронювати</Button>
      </Card>
    </div>
  );
};
```

2. Настроюваність:

- Використання вбудованих класів та стилів для налаштування компонентів відповідно до дизайну вашого проекту:

jsx

```
<Button className="bg-green-500 hover:bg-green-700 text-white">
  Забронювати
</Button>
```

Висновок

Розробка сайту за допомогою сучасних технологій, таких як Next.js, Tailwind CSS, Zustand та UI бібліотеки shadcn, дозволяє створити високопродуктивний, адаптивний та зручний у використанні веб-ресурс. Кожна з цих технологій має свої унікальні переваги, які разом забезпечують ефективну розробку, привабливий дизайн та інтуїтивно зрозумілий інтерфейс для кінцевих користувачів.

Розділ 5: Інтеграція контенту

У цьому розділі розглянемо ключові аспекти інтеграції контенту на веб-сайт для надання туристичних послуг. Це включає процес збору та підготовки контенту, завантаження його на сайт і організацію для забезпечення зручності користувачів та ефективної роботи сайту.

5.1 Збір і підготовка контенту

Збір контенту є першим кроком у процесі інтеграції і включає кілька етапів:

1. Визначення типів контенту:

- Тексти: Опис турів, готелів, місць призначення, статті в блозі та новини.
- Зображення: Фотографії місць призначення, турів, готелів, а також ілюстрації для статей.
- Відео: Промо-ролики, відео-огляди турів і місць призначення.
- Відгуки: Текстові відгуки клієнтів та рейтинги.

2. Збір інформації:

- Власні ресурси: Інформація, створена внутрішньою командою компанії.
- Партнери: Інформація від партнерів, таких як готелі, туристичні агенції та туроператори.
- Користувачі: Контент, створений користувачами, включаючи відгуки та фотографії.

3. Редагування та підготовка:

- Редагування тексту: Перевірка орфографії, граматики, стилістики та узгодження інформації.
- Оптимізація зображень: Редагування фотографій для зменшення розміру файлів без втрати якості, додавання альт-тегів для SEO.

- Обробка відео: Монтаж відео для створення професійного вигляду, оптимізація для веб-публікації.
- Форматування відгуків: Структурування відгуків у зручному форматі, виділення ключових моментів.

5.2 Завантаження контенту на сайт

Завантаження контенту включає розміщення підготовлених матеріалів на веб-сайті за допомогою системи управління контентом (CMS) або інших інструментів.

1. Використання CMS:
   - WordPress: Використання плагінів для завантаження та організації контенту.
   - Інші CMS: Використання систем, які підтримують сучасні технології, такі як Next.js.
2. Розміщення текстів:
   - Статті та блоги: Додавання текстів у відповідні розділи сайту.
   - Опис турів і готелів: Введення детальної інформації в картки продуктів або сторінки опису.
3. Завантаження зображень і відео:
   - Медіа-бібліотека: Створення організованої бібліотеки зображень та відео для полегшення доступу та використання.
   - Галереї та слайдери: Використання галерей для показу фотографій та відео у зручному форматі.
4. Інтеграція відгуків:
   - Форми для відгуків: Створення форм для збору відгуків від користувачів.
   - Відображення відгуків: Використання віджетів для відображення відгуків на сторінках турів, готелів та інших послуг.

5.3 Організація контенту

Організація контенту забезпечує легкий доступ користувачів до інформації та покращує загальний досвід користування сайтом.

1. Категоризація:

- Розділи сайту: Створення чітких розділів для різних типів контенту, таких як тури, готелі, блог, відгуки.
- Теги та категорії: Використання тегів та категорій для організації статей та відгуків, що полегшує пошук інформації.

2. Навігація:

- Меню: Створення інтуїтивно зрозумілого меню, яке дозволяє легко переходити між основними розділами сайту.
- Пошук: Інтеграція пошукової функції для швидкого знаходження потрібної інформації.

3. Структура сторінок:

- Шаблони сторінок: Використання шаблонів для уніфікації вигляду сторінок турів, готелів, статей та відгуків.
- Інтерактивні елементи: Додавання інтерактивних елементів, таких як карти, слайдери та віджети для покращення взаємодії користувачів із контентом.

4. Оптимізація:

- Швидкість завантаження: Оптимізація розміру зображень і відео для зменшення часу завантаження сторінок.
- SEO: Використання мета-тегів, ключових слів, альт-тегів для зображень та оптимізованих URL-адрес для покращення видимості в пошукових системах.

Висновок

Інтеграція контенту є важливим етапом у розробці сайту для надання туристичних послуг. Збір і підготовка контенту, завантаження його на сайт та організація забезпечують якісну презентацію інформації, полегшують

доступ користувачів до необхідних даних та покращують загальний досвід користування сайтом. Використання сучасних інструментів і методів дозволяє створити ефективний і привабливий веб-ресурс, який відповідає потребам і очікуванням цільової аудиторії.

Розділ 6: Тестування сайту

Тестування є важливим етапом у розробці веб-сайту, який дозволяє виявити та виправити можливі помилки, забезпечити коректну роботу на різних пристроях та браузерах, і забезпечити позитивний користувацький досвід. У цьому розділі розглянемо основні аспекти тестування сайту, включаючи тестування на різних пристроях, у різних браузерах, а також процес виправлення помилок.

6.1 Тестування на різних пристроях

Тестування на різних пристроях забезпечує коректну роботу сайту на різноманітних девайсах, включаючи десктопи, ноутбуки, планшети та смартфони.

1. Важливість тестування на різних пристроях:

- Забезпечення адаптивності: Перевірка коректного відображення та функціонування сайту на екранах різних розмірів.
- Користувацький досвід: Забезпечення зручності користування незалежно від типу пристрою.

2. Методи тестування:

- Емулюючі інструменти: Використання браузерних інструментів розробника для емулювання різних пристроїв (наприклад, Chrome DevTools).
- Реальні пристрої: Тестування на фізичних пристроях для більш точного виявлення проблем.

3. Ключові аспекти тестування:

- Навігація: Перевірка зручності навігації на маленьких екранах.
- Інтерактивність: Перевірка коректної роботи інтерактивних елементів (кнопок, форм, меню).
- Швидкість завантаження: Оцінка швидкості завантаження сторінок на різних пристроях.

6.2 Тестування в різних браузерах

Тестування в різних браузерах гарантує, що сайт коректно відображається та функціонує у всіх популярних веб-браузерах.

1. Важливість кросбраузерного тестування:

- Забезпечення сумісності: Врахування різних особливостей рендерингу та виконання скриптів у різних браузерах.
- Поширеність: Популярні браузери включають Google Chrome, Mozilla Firefox, Safari, Microsoft Edge та інші.

2. Методи тестування:

- Ручне тестування: Перевірка сайту у всіх необхідних браузерах вручну.
- Автоматизоване тестування: Використання інструментів для автоматизованого кросбраузерного тестування (наприклад, BrowserStack, CrossBrowserTesting).

3. Ключові аспекти тестування:

- Сумісність стилів: Перевірка коректного відображення стилів (CSS) у різних браузерах.
- Виконання скриптів: Перевірка коректної роботи JavaScript-функціоналу.
- Особливості рендерингу: Виявлення відмінностей у рендерингу HTML та CSS у різних браузерах.

6.3 Виправлення помилок

Виправлення помилок є критичним етапом, який включає ідентифікацію, аналіз та усунення проблем, виявлених під час тестування.

1. Процес виправлення помилок:

- Виявлення помилок: Збір інформації про виявлені проблеми під час тестування.

- Аналіз: Визначення причин помилок та розробка стратегії їх усунення.
- Виправлення: Внесення необхідних змін у код та повторне тестування для підтвердження виправлення.

2. Інструменти для виправлення помилок:

- Консоль розробника: Використання браузерних інструментів для виявлення помилок JavaScript, CSS та мережевих запитів.
- Дебагери: Використання інструментів для відлагодження коду (наприклад, VSCode, WebStorm).

3. Документування:

- Логування помилок: Ведення журналу виявлених та виправлених помилок.
- Репорти: Створення звітів про виконане тестування та внесені зміни.

Висновок

Тестування сайту на різних пристроях та у різних браузерах є важливим кроком для забезпечення його стабільної та коректної роботи. Виявлення та виправлення помилок дозволяє покращити користувацький досвід і забезпечити сумісність з різними технологіями. Завдяки ретельному тестуванню та налагодженню веб-сайт для надання туристичних послуг буде відповідати високим стандартам якості та задовольняти потреби користувачів.

Розділ 7: Результати проекту

У цьому розділі підсумовуються результати розробки сайту для надання послуг у сфері туризму, досягнуті цілі, а також оцінюється успішність проекту. Розглядаються ключові аспекти виконаних робіт, їх вплив на бізнес та користувачів, а також подальші кроки для вдосконалення проекту.

7.1 Досягнуті цілі

1. Створення сучасного сайту:

- Використання сучасних технологій, таких як Next.js, Tailwind CSS, Zustand та shadcn UI, дозволило створити високопродуктивний і адаптивний сайт.
- Сайт відповідає вимогам щодо швидкості завантаження, сумісності з різними пристроями та браузерами.

2. Інтуїтивно зрозумілий інтерфейс:

- Розробка користувацького інтерфейсу, що забезпечує зручну навігацію та взаємодію з контентом.
- Вибір кольорової гами і шрифтів, які відповідають брендінгу компанії та створюють приємний візуальний досвід.

3. Інтеграція контенту:

- Збір і підготовка якісного контенту, включаючи тексти, зображення, відео та відгуки, які надають користувачам повну інформацію про туристичні послуги.
- Організація контенту на сайті, що полегшує доступ користувачів до необхідної інформації.

4. Тестування і виправлення помилок:

- Тестування сайту на різних пристроях і браузерах для забезпечення стабільної роботи.

- Виправлення виявлених помилок і оптимізація продуктивності сайту.

7.2 Вплив на бізнес

1. Покращення користувацького досвіду:

- Зручний і функціональний сайт підвищує задоволеність користувачів і спонукає їх частіше використовувати послуги компанії.
- Легкість навігації та швидкий доступ до інформації сприяють зростанню конверсій і продажів.

2. Підвищення видимості в Інтернеті:

- Оптимізація для пошукових систем (SEO) дозволяє покращити видимість сайту в пошукових результатах, залучаючи більше органічного трафіку.
- Використання сучасних веб-технологій покращує продуктивність сайту і позитивно впливає на його рейтинг у пошукових системах.

3. Ефективне управління контентом:

- Використання CMS та інших інструментів для зручного управління контентом дозволяє швидко оновлювати інформацію та додавати новий контент.
- Це забезпечує актуальність інформації на сайті та оперативне реагування на зміни у пропозиціях туристичних послуг.

7.3 Показники успішності проекту

1. Аналітика та метрики:

- Відстеження показників відвідуваності сайту, тривалості сеансів, показників відмов та конверсій.
- Аналізуючи ці дані, можна оцінити ефективність сайту та виявити можливі напрямки для покращення.

2. Зворотний зв'язок від користувачів:

- Збір відгуків від користувачів щодо зручності користування сайтом, його дизайну та функціональності.
- Використання цього зворотного зв'язку для подальшого вдосконалення сайту.

3. Бізнес-результати:

- Оцінка зростання продажів та кількості замовлень туристичних послуг через сайт.
- Вимірювання впливу на брендінг та репутацію компанії.

7.4 Подальші кроки

1. Подальший розвиток функціональності:

- Додавання нових функцій та інтерактивних елементів для покращення користувацького досвіду.
- Розширення можливостей персоналізації контенту для користувачів.

2. Підтримка та оновлення:

- Регулярні оновлення сайту для забезпечення його безпеки та стабільної роботи.
- Оновлення контенту та дизайну відповідно до змін у потребах користувачів та ринку.

3. Маркетинг і промоція:

- Використання різних маркетингових стратегій для залучення нових користувачів на сайт.
- Активне просування сайту через соціальні мережі, SEO та контент-маркетинг.

Висновок

Проект розробки сайту для надання туристичних послуг досягнув своїх цілей, створивши сучасний, адаптивний та функціональний веб-ресурс. Завдяки використанню сучасних технологій, якісного контенту та ретельного тестування, сайт забезпечує позитивний користувацький досвід,

сприяє зростанню бізнесу та підвищує задоволеність користувачів. Подальший розвиток і підтримка сайту дозволять зберігати його актуальність і ефективність у динамічному середовищі туристичної індустрії.

Висновки і рекомендації

У цьому розділі підводяться підсумки реалізації проекту, оцінюються досягнення та визначаються напрямки подальшого вдосконалення. Висновки відображають основні результати проекту, а рекомендації включають поради для майбутнього розвитку та підтримки сайту.

Висновки

1. Успішна реалізація проекту:

- Проект створення сайту для надання туристичних послуг був успішно реалізований із використанням сучасних технологій Next.js, Tailwind CSS, Zustand і shadcn UI.
- Сайт відповідає всім заявленим вимогам щодо функціональності, адаптивності та зручності користування.

2. Покращення користувацького досвіду:

- Розробка інтуїтивно зрозумілого інтерфейсу та інтеграція якісного контенту забезпечили позитивний користувацький досвід.
- Оптимізація сайту для мобільних пристроїв та різних браузерів підвищила доступність та зручність використання.

3. Висока продуктивність і SEO-оптимізація:

- Використання сучасних веб-технологій дозволило досягти високої швидкості завантаження сторінок і покращити SEO-показники.
- Сайт успішно інтегрує контент із застосуванням мета-тегів, ключових слів та оптимізованих URL-адрес.

4. Ефективне управління контентом:

- Використання CMS та інших інструментів дозволило зручно збирати, підготовляти та організовувати контент на сайті.
- Це забезпечило актуальність інформації та швидке оновлення контенту.

Рекомендації

1. Подальший розвиток функціональності:

- Додавання нових інтерактивних елементів, таких як інтегровані карти, чат-боти для підтримки користувачів та інструменти для бронювання послуг онлайн.
- Розширення можливостей персоналізації контенту для покращення взаємодії з користувачами.

2. Підтримка та регулярні оновлення:

- Проведення регулярних технічних оновлень для забезпечення безпеки та стабільної роботи сайту.
- Постійне оновлення контенту відповідно до нових пропозицій, сезонних акцій та змін у туристичній індустрії.

3. Аналіз та покращення користувацького досвіду:

- Збір та аналіз відгуків від користувачів для виявлення слабких місць та їх усунення.
- Використання аналітичних інструментів для відстеження поведінки користувачів на сайті та оптимізації взаємодії.

4. Розширення маркетингових стратегій:

- Активне використання соціальних мереж для просування сайту та залучення нових користувачів.
- Інвестування в SEO та контент-маркетинг для покращення видимості сайту у пошукових системах.

5. Моніторинг та реагування на тенденції ринку:

- Відстеження нових тенденцій у туристичній індустрії та адаптація сайту відповідно до змін у ринку.
- Впровадження нових технологій та функціональності для збереження конкурентоспроможності.

Заключне слово

Проект створення сайту для надання послуг у сфері туризму успішно досяг своїх цілей, забезпечивши високий рівень якості та задоволення користувачів. Подальший розвиток та підтримка сайту дозволять зберігати його актуальність і ефективність, сприяючи зростанню бізнесу та задоволенню потреб клієнтів. Виконання рекомендацій допоможе покращити функціональність сайту, розширити його можливості та підтримувати високу якість послуг, що надаються.

Додатки

eclare namespace QuickLRU {

interface Options<KeyType, ValueType> {

/**

The maximum number of milliseconds an item should remain in the cache.

@default Infinity

By default, `maxAge` will be `Infinity`, which means that items will never expire.

Lazy expiration upon the next write or read call.

Individual expiration of an item can be specified by the `set(key, value, maxAge)` method.

*/

readonly maxAge?: number;

/**

The maximum number of items before evicting the least recently used items.

*/

readonly maxSize: number;

/**

Called right before an item is evicted from the cache.

Useful for side effects or for items like object URLs that need explicit cleanup (`revokeObjectURL`).

*/

onEviction?: (key: KeyType, value: ValueType) => void;

}

```
}

declare class QuickLRU<KeyType, ValueType>
implements Iterable<[KeyType, ValueType]> {
/**
The stored item count.
*/
readonly size: number;

/**
Simple ["Least Recently Used" (LRU)
cache](https://en.m.wikipedia.org/wiki/Cache_replacement_policies#Least_Rec
ently_Used_.28LRU.29).

The instance is
[`iterable`](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Iter
ation_protocols) so you can use it directly in a
[`for...of`](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Stat
ements/for...of) loop.

@example
```
import QuickLRU = require('quick-lru');

const lru = new QuickLRU({maxSize: 1000});
```
```

lru.set('🐴', '🌈');

lru.has('🐴');
//=> true

lru.get('🐴');
//=> '🌈'
```
*/
constructor(options: QuickLRU.Options<KeyType, ValueType>);

[Symbol.iterator](): IterableIterator<[KeyType, ValueType]>;

/**
Set an item. Returns the instance.

Individual expiration of an item can be specified with the `maxAge` option. If not specified, the global `maxAge` value will be used in case it is specified in the constructor, otherwise the item will never expire.

@returns The list instance.
*/
set(key: KeyType, value: ValueType, options?: {maxAge?: number}): this;
```

```
/**
Get an item.

@returns The stored item or `undefined`.
*/
get(key: KeyType): ValueType | undefined;

/**
Check if an item exists.
*/
has(key: KeyType): boolean;

/**
Get an item without marking it as recently used.

@returns The stored item or `undefined`.
*/
peek(key: KeyType): ValueType | undefined;

/**
Delete an item.

@returns `true` if the item is removed or `false` if the item doesn't exist.
*/
```

```
delete(key: KeyType): boolean;

/**
Delete all items.
*/
clear(): void;

/**
Update the `maxSize` in-place, discarding items as necessary. Insertion order is mostly preserved, though this is not a strong guarantee.

Useful for on-the-fly tuning of cache sizes in live systems.
*/
resize(maxSize: number): void;

/**
Iterable for all the keys.
*/
keys(): IterableIterator<KeyType>;

/**
Iterable for all the values.
*/
values(): IterableIterator<ValueType>;
```

```
/**
Iterable for all entries, starting with the oldest (ascending in recency).
*/
entriesAscending(): IterableIterator<[KeyType, ValueType]>;

/**
Iterable for all entries, starting with the newest (descending in recency).
*/
entriesDescending(): IterableIterator<[KeyType, ValueType]>;
}

export = QuickLRU;
"name": "call-bind",
"version": "1.0.7",
"description": "Robustly `.call.bind()` a function",
"main": "index.js",
"exports": {
".": "./index.js",
"./callBound": "./callBound.js",
"./package.json": "./package.json"
},
"scripts": {
"prepack": "npmignore --auto --commentLines=auto",
"prepublish": "not-in-publish || npm run prepublishOnly",
"prepublishOnly": "safe-publish-latest",
```

"lint": "eslint --ext=.js,.mjs .",
"postlint": "evalmd README.md",
"pretest": "npm run lint",
"tests-only": "nyc tape 'test/**/*.js'",
"test": "npm run tests-only",
"posttest": "aud --production",
"version": "auto-changelog && git add CHANGELOG.md",
"postversion": "auto-changelog && git add CHANGELOG.md && git commit --no-edit --amend && git tag -f \"v$(node -e \"console.log(require('./package.json').version)\")\""
},
"repository": {
"type": "git",
"url": "git+https://github.com/ljharb/call-bind.git"
},
"keywords": [
"javascript",
"ecmascript",
"es",
"js",
"callbind",
"callbound",
"call",
"bind",
"bound",

"call-bind",
"call-bound",
"function",
"es-abstract"
],
"author": "Jordan Harband <ljharb@gmail.com>",
"funding": {
"url": "https://github.com/sponsors/ljharb"
},
"license": "MIT",
"bugs": {
"url": "https://github.com/ljharb/call-bind/issues"
},
"homepage": "https://github.com/ljharb/call-bind#readme",
"devDependencies": {
"@ljharb/eslint-config": "^21.1.0",
"aud": "^2.0.4",
"auto-changelog": "^2.4.0",
"es-value-fixtures": "^1.4.2",
"eslint": "=8.8.0",
"evalmd": "^0.0.19",
"for-each": "^0.3.3",
"gopd": "^1.0.1",
"has-strict-mode": "^1.0.1",
"in-publish": "^2.0.1",

"npmignore": "^0.3.1",
"nyc": "^10.3.2",
"object-inspect": "^1.13.1",
"safe-publish-latest": "^2.0.0",
"tape": "^5.7.4"
},
"dependencies": {
"es-define-property": "^1.0.0",
"es-errors": "^1.3.0",
"function-bind": "^1.1.2",
"get-intrinsic": "^1.2.4",
"set-function-length": "^1.2.1"
},
"testling": {
"files": "test/index.js"
},
"auto-changelog": {
"output": "CHANGELOG.md",
"template": "keepachangelog",
"unreleased": false,
"commitLimit": false,
"backfillLimit": false,
"hideCredit": true
},
"publishConfig": {

```
"ignore": [
".github/workflows"
]
},
"engines": {
"node": ">= 0.4"
}
}
/ File: C:\Users\dimas\Desktop\dima_diplom_mine\app\src\app\layout.tsx
import * as entry from '../../../src/app/layout.js'
import type { ResolvingMetadata, ResolvingViewport } from
'next/dist/lib/metadata/types/metadata-interface.js'

type TEntry = typeof import('../../../src/app/layout.js')

// Check that the entry is a valid entry
checkFields<Diff<{
default: Function
config?: {}
generateStaticParams?: Function
revalidate?: RevalidateRange<TEntry> | false
dynamic?: 'auto' | 'force-dynamic' | 'error' | 'force-static'
dynamicParams?: boolean
fetchCache?: 'auto' | 'force-no-store' | 'only-no-store' | 'default-no-store' |
'default-cache' | 'only-cache' | 'force-cache'
```

```
  preferredRegion?: 'auto' | 'global' | 'home' | string | string[]
  runtime?: 'nodejs' | 'experimental-edge' | 'edge'
  maxDuration?: number

  metadata?: any
  generateMetadata?: Function
  viewport?: any
  generateViewport?: Function

  }, TEntry, ">>()

  // Check the prop type of the entry function
  checkFields<Diff<LayoutProps, FirstArg<TEntry['default']>, 'default'>>()

  // Check the arguments and return type of the generateMetadata function
  if ('generateMetadata' in entry) {
  checkFields<Diff<LayoutProps, FirstArg<MaybeField<TEntry,
'generateMetadata'>>, 'generateMetadata'>>()
  checkFields<Diff<ResolvingMetadata, SecondArg<MaybeField<TEntry,
'generateMetadata'>>, 'generateMetadata'>>()
  }

  // Check the arguments and return type of the generateViewport function
  if ('generateViewport' in entry) {
```

```
        checkFields<Diff<LayoutProps, FirstArg<MaybeField<TEntry,
'generateViewport'>>, 'generateViewport'>>()
        checkFields<Diff<ResolvingViewport, SecondArg<MaybeField<TEntry,
'generateViewport'>>, 'generateViewport'>>()
        }

        // Check the arguments and return type of the generateStaticParams
function
        if ('generateStaticParams' in entry) {
        checkFields<Diff<{ params: PageParams },
FirstArg<MaybeField<TEntry, 'generateStaticParams'>>,
'generateStaticParams'>>()
        checkFields<Diff<{ __tag__: 'generateStaticParams', __return_type__:
any[] | Promise<any[]> }, { __tag__: 'generateStaticParams', __return_type__:
ReturnType<MaybeField<TEntry, 'generateStaticParams'>> }>>()
        }

        type PageParams = any
        export interface PageProps {
        params?: any
        searchParams?: any
        }
        export interface LayoutProps {
        children?: React.ReactNode
```

```
params?: any
}

// ==========
// Utility types
type RevalidateRange<T> = T extends { revalidate: any } ?
NonNegative<T['revalidate']> : never

// If T is unknown or any, it will be an empty {} type. Otherwise, it will be
the same as Omit<T, keyof Base>.
type OmitWithTag<T, K extends keyof any, _M> = Omit<T, K>
type Diff<Base, T extends Base, Message extends string = ''> = 0 extends
(1 & T) ? {} : OmitWithTag<T, keyof Base, Message>

type FirstArg<T extends Function> = T extends (...args: [infer T, any]) =>
any ? unknown extends T ? any : T : never
type SecondArg<T extends Function> = T extends (...args: [any, infer T])
=> any ? unknown extends T ? any : T : never
type MaybeField<T, K extends string> = T extends { [k in K]: infer G } ?
G extends Function ? G : never : never

function checkFields<_ extends { [k in keyof any]: never }>() {}

// https://github.com/sindresorhus/type-fest
```

```
type Numeric = number | bigint

type Zero = 0 | 0n

type Negative<T extends Numeric> = T extends Zero ? never : `${T}`
extends `-${string}` ? T : never

type NonNegative<T extends Numeric> = T extends Zero ? T :
Negative<T> extends never ? T : '__invalid_negative_number__'
```

```
/*!*****************************************************************
*****************************************************************
*****************************************************************
*****************************************************************
*****************************************************************
*****************************************************************
*****************************!*\
!*** css ./node_modules/next/dist/build/webpack/loaders/css-
loader/src/index.js??ruleSet[1].rules[13].oneOf[2].use[1]!./node_modules/next/d
ist/build/webpack/loaders/next-font-
loader/index.js??ruleSet[1].rules[13].oneOf[2].use[2]!./node_modules/next/font/
google/target.css?{"path":"src\\app\\layout.tsx","import":"Inter","arguments":[{"
subsets":["latin"],"variable":"--font-sans"}],"variableName":"inter"} ***!
\*****************************************************************
*****************************************************************
*****************************************************************
*****************************************************************
*****************************************************************
*****************************************************************
*****************************/
/* cyrillic-ext */
@font-face {
font-family: '__Inter_aaf875';
font-style: normal;
font-weight: 100 900;
font-display: swap;
```

```
src: url(/_next/static/media/ec159349637c90ad-s.woff2) format('woff2');
unicode-range: U+0460-052F, U+1C80-1C88, U+20B4, U+2DE0-2DFF,
U+A640-A69F, U+FE2E-FE2F;
}
/* cyrillic */
@font-face {
font-family: '__Inter_aaf875';
font-style: normal;
font-weight: 100 900;
font-display: swap;
src: url(/_next/static/media/513657b02c5c193f-s.woff2) format('woff2');
unicode-range: U+0301, U+0400-045F, U+0490-0491, U+04B0-04B1,
U+2116;
}
/* greek-ext */
@font-face {
font-family: '__Inter_aaf875';
font-style: normal;
font-weight: 100 900;
font-display: swap;
src: url(/_next/static/media/fd4db3eb5472fc27-s.woff2) format('woff2');
unicode-range: U+1F00-1FFF;
}
/* greek */
@font-face {
```

```
font-family: '__Inter_aaf875';
font-style: normal;
font-weight: 100 900;
font-display: swap;
src: url(/_next/static/media/51ed15f9841b9f9d-s.woff2) format('woff2');
unicode-range: U+0370-0377, U+037A-037F, U+0384-038A, U+038C,
U+038E-03A1, U+03A3-03FF;
}
/* vietnamese */
@font-face {
font-family: '__Inter_aaf875';
font-style: normal;
font-weight: 100 900;
font-display: swap;
src: url(/_next/static/media/05a31a2ca4975f99-s.woff2) format('woff2');
unicode-range: U+0102-0103, U+0110-0111, U+0128-0129, U+0168-
0169, U+01A0-01A1, U+01AF-01B0, U+0300-0301, U+0303-0304, U+0308-
0309, U+0323, U+0329, U+1EA0-1EF9, U+20AB;

/*
1. Use a consistent sensible line-height in all browsers.
2. Prevent adjustments of font size after orientation changes in iOS.
3. Use a more readable tab size.
4. Use the user's configured `sans` font-family by default.
5. Use the user's configured `sans` font-feature-settings by default.
```

```
6. Use the user's configured `sans` font-variation-settings by default.
7. Disable tap highlights on iOS
*/

html,
:host {
line-height: 1.5; /* 1 */
-webkit-text-size-adjust: 100%; /* 2 */
-moz-tab-size: 4; /* 3 */
tab-size: 4; /* 3 */
font-family: var(--font-sans), ui-sans-serif, system-ui, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji"; /* 4 */
font-feature-settings: normal; /* 5 */
font-variation-settings: normal; /* 6 */
-webkit-tap-highlight-color: transparent; /* 7 */
}

/*
1. Remove the margin in all browsers.
2. Inherit line-height from `html` so users can set them as a class directly on the `html` element.
*/

body {
```

```
margin: 0; /* 1 */
line-height: inherit; /* 2 */
}

/*
1. Add the correct height in Firefox.
2. Correct the inheritance of border color in Firefox.
(https://bugzilla.mozilla.org/show_bug.cgi?id=190655)
3. Ensure horizontal rules are visible by default.
*/

hr {
height: 0; /* 1 */
color: inherit; /* 2 */
border-top-width: 1px; /* 3 */
}

/*
Add the correct text decoration in Chrome, Edge, and Safari.
*/

abbr:where([title]) {
text-decoration: underline dotted;
}
```

```
/*
Remove the default font size and weight for headings.
*/

h1,
h2,
h3,
h4,
h5,
h6 {
font-size: inherit;
font-weight: inherit;
}

/*
Reset links to optimize for opt-in styling instead of opt-out.
*/

a {
color: inherit;
text-decoration: inherit;
}

/*
Add the correct font weight in Edge and Safari.
```

```
*/

b,
strong {
font-weight: bolder;
}

/*
1. Use the user's configured `mono` font-family by default.
2. Use the user's configured `mono` font-feature-settings by default.
3. Use the user's configured `mono` font-variation-settings by default.
4. Correct the odd `em` font sizing in all browsers.
*/

code,
kbd,
samp,
pre {
font-family: ui-monospace, SFMono-Regular, Menlo, Monaco, Consolas,
"Liberation Mono", "Courier New", monospace; /* 1 */
font-feature-settings: normal; /* 2 */
font-variation-settings: normal; /* 3 */
font-size: 1em; /* 4 */
}
```

```
/*
Add the correct font size in all browsers.
*/

small {
font-size: 80%;
}

/*
Prevent `sub` and `sup` elements from affecting the line height in all browsers.
*/

sub,
sup {
font-size: 75%;
line-height: 0;
position: relative;
vertical-align: baseline;
}

sub {
bottom: -0.25em;
}
```

```
sup {
top: -0.5em;
}

/*
1. Remove text indentation from table contents in Chrome and Safari.
(https://bugs.chromium.org/p/chromium/issues/detail?id=999088,
https://bugs.webkit.org/show_bug.cgi?id=201297)
2. Correct table border color inheritance in all Chrome and Safari.
(https://bugs.chromium.org/p/chromium/issues/detail?id=935729,
https://bugs.webkit.org/show_bug.cgi?id=195016)
3. Remove gaps between table borders by default.
*/

table {
text-indent: 0; /* 1 */
border-color: inherit; /* 2 */
border-collapse: collapse; /* 3 */
}

/*
1. Change the font styles in all browsers.
2. Remove the margin in Firefox and Safari.
3. Remove default padding in all browsers.
*/
```

```
button,
input,
optgroup,
select,
textarea {
font-family: inherit; /* 1 */
font-feature-settings: inherit; /* 1 */
font-variation-settings: inherit; /* 1 */
font-size: 100%; /* 1 */
font-weight: inherit; /* 1 */
line-height: inherit; /* 1 */
letter-spacing: inherit; /* 1 */
color: inherit; /* 1 */
margin: 0; /* 2 */
padding: 0; /* 3 */
}

/*
Remove the inheritance of text transform in Edge and Firefox.
*/

button,
select {
text-transform: none;
```

```
}

/*
1. Correct the inability to style clickable types in iOS and Safari.
2. Remove default button styles.
*/

button,
input:where([type='button']),
input:where([type='reset']),
input:where([type='submit']) {
-webkit-appearance: button; /* 1 */
background-color: transparent; /* 2 */
background-image: none; /* 2 */
}

/*
Use the modern Firefox focus style for all focusable elements.
*/

:-moz-focusring {
outline: auto;
}

/*
```

```
        Remove the additional `:invalid` styles in Firefox.
(https://github.com/mozilla/gecko-
dev/blob/2f9eacd9d3d995c937b4251a5557d95d494c9be1/layout/style/res/forms
.css#L728-L737)
        */

        :-moz-ui-invalid {
        box-shadow: none;
        }

        /*
        Add the correct vertical alignment in Chrome and Firefox.
        */

        progress {
        vertical-align: baseline;
        }

        /*
        Correct the cursor style of increment and decrement buttons in Safari.
        */

        ::-webkit-inner-spin-button,
        ::-webkit-outer-spin-button {
        height: auto;
```

```
}

/*
1. Correct the odd appearance in Chrome and Safari.
2. Correct the outline style in Safari.
*/

[type='search'] {
-webkit-appearance: textfield; /* 1 */
outline-offset: -2px; /* 2 */
}

/*
Remove the inner padding in Chrome and Safari on macOS.
*/

::-webkit-search-decoration {
-webkit-appearance: none;
}

/*
1. Correct the inability to style clickable types in iOS and Safari.
2. Change font properties to `inherit` in Safari.
*/
```

```
::-webkit-file-upload-button {
-webkit-appearance: button; /* 1 */
font: inherit; /* 2 */
}

/*
Add the correct display in Chrome and Safari.
*/

summary {
display: list-item;
}

/*
Removes the default spacing and border for appropriate elements.
*/

blockquote,
dl,
dd,
h1,
h2,
h3,
h4,
h5,
```

```
h6,
hr,
figure,
p,
pre {
margin: 0;
}

fieldset {
margin: 0;
padding: 0;
}

legend {
padding: 0;
}

ol,
ul,
menu {
list-style: none;
margin: 0;
padding: 0;
}
```

```
/*
Reset default styling for dialogs.
*/
dialog {
padding: 0;
}

/*
Prevent resizing textareas horizontally by default.
*/

textarea {
resize: vertical;
}

/*
1. Reset the default placeholder opacity in Firefox.
(https://github.com/tailwindlabs/tailwindcss/issues/3300)
2. Set the default placeholder color to the user's configured gray 400
color.
*/

input::placeholder,
textarea::placeholder {
opacity: 1; /* 1 */
```

```
color: #9ca3af; /* 2 */
}

/*
Set the default cursor for buttons.
*/

button,
[role="button"] {
cursor: pointer;
}

/*
Make sure disabled buttons don't get the pointer cursor.
*/
:disabled {
cursor: default;
}

/*
1. Make replaced elements `display: block` by default.
(https://github.com/mozdevs/cssremedy/issues/14)
2. Add `vertical-align: middle` to align replaced elements more sensibly
by default. (https://github.com/jensimmons/cssremedy/issues/14#issuecomment-
634934210)
```

```
This can trigger a poorly considered lint error in some tools but is
included by design.
*/

img,
svg,
video,
canvas,
audio,
iframe,
embed,
object {
display: block; /* 1 */
vertical-align: middle; /* 2 */
}

/*
Constrain images and videos to the parent width and preserve their
intrinsic aspect ratio. (https://github.com/mozdevs/cssremedy/issues/14)
*/

img,
video {
max-width: 100%;
height: auto;
```

```
}

/* Make elements with the HTML hidden attribute stay hidden by default
*/
[hidden] {
display: none;
}
:root {
--background: 0 0% 100%;
--foreground: 222.2 84% 4.9%;

--card: 0 0% 100%;
--card-foreground: 222.2 84% 4.9%;

--popover: 0 0% 100%;
--popover-foreground: 222.2 84% 4.9%;

--primary: 222.2 47.4% 11.2%;
--primary-foreground: 210 40% 98%;

--secondary: 210 40% 96.1%;
--secondary-foreground: 222.2 47.4% 11.2%;

--muted: 210 40% 96.1%;
--muted-foreground: 215.4 16.3% 46.9%;
```

```
--accent: 210 40% 96.1%;
--accent-foreground: 222.2 47.4% 11.2%;

--destructive: 0 84.2% 60.2%;
--destructive-foreground: 210 40% 98%;

--border: 214.3 31.8% 91.4%;
--input: 214.3 31.8% 91.4%;
--ring: 222.2 84% 4.9%;

--radius: 0.5rem;
}
* {
border-color: hsl(var(--border));
}
body {
background-color: hsl(var(--background));
color: hsl(var(--foreground));
}

*, ::before, ::after {
--tw-border-spacing-x: 0;
--tw-border-spacing-y: 0;
--tw-translate-x: 0;
```

```
--tw-translate-y: 0;
--tw-rotate: 0;
--tw-skew-x: 0;
--tw-skew-y: 0;
--tw-scale-x: 1;
--tw-scale-y: 1;
--tw-pan-x: ;
--tw-pan-y: ;
--tw-pinch-zoom: ;
--tw-scroll-snap-strictness: proximity;
--tw-gradient-from-position: ;
--tw-gradient-via-position: ;
--tw-gradient-to-position: ;
--tw-ordinal: ;
--tw-slashed-zero: ;
--tw-numeric-figure: ;
--tw-numeric-spacing: ;
--tw-numeric-fraction: ;
--tw-ring-inset: ;
--tw-ring-offset-width: 0px;
--tw-ring-offset-color: #fff;
--tw-ring-color: rgb(59 130 246 / 0.5);
--tw-ring-offset-shadow: 0 0 #0000;
--tw-ring-shadow: 0 0 #0000;
--tw-shadow: 0 0 #0000;
```

```
--tw-shadow-colored: 0 0 #0000;
--tw-blur: ;
--tw-brightness: ;
--tw-contrast: ;
--tw-grayscale: ;
--tw-hue-rotate: ;
--tw-invert: ;
--tw-saturate: ;
--tw-sepia: ;
--tw-drop-shadow: ;
--tw-backdrop-blur: ;
--tw-backdrop-brightness: ;
--tw-backdrop-contrast: ;
--tw-backdrop-grayscale: ;
--tw-backdrop-hue-rotate: ;
--tw-backdrop-invert: ;
--tw-backdrop-opacity: ;
--tw-backdrop-saturate: ;
--tw-backdrop-sepia: ;
--tw-contain-size: ;
--tw-contain-layout: ;
--tw-contain-paint: ;
--tw-contain-style: ;
}
```

```
::backdrop {
--tw-border-spacing-x: 0;
--tw-border-spacing-y: 0;
--tw-translate-x: 0;
--tw-translate-y: 0;
--tw-rotate: 0;
--tw-skew-x: 0;
--tw-skew-y: 0;
--tw-scale-x: 1;
--tw-scale-y: 1;
--tw-pan-x: ;
--tw-pan-y: ;
--tw-pinch-zoom: ;
--tw-scroll-snap-strictness: proximity;
--tw-gradient-from-position: ;
--tw-gradient-via-position: ;
--tw-gradient-to-position: ;
--tw-ordinal: ;
--tw-slashed-zero: ;
--tw-numeric-figure: ;
--tw-numeric-spacing: ;
--tw-numeric-fraction: ;
--tw-ring-inset: ;
--tw-ring-offset-width: 0px;
--tw-ring-offset-color: #fff;
```

```
--tw-ring-color: rgb(59 130 246 / 0.5);
--tw-ring-offset-shadow: 0 0 #0000;
--tw-ring-shadow: 0 0 #0000;
--tw-shadow: 0 0 #0000;
--tw-shadow-colored: 0 0 #0000;
--tw-blur: ;
--tw-brightness: ;
--tw-contrast: ;
--tw-grayscale: ;
--tw-hue-rotate: ;
--tw-invert: ;
--tw-saturate: ;
--tw-sepia: ;
--tw-drop-shadow: ;
--tw-backdrop-blur: ;
--tw-backdrop-brightness: ;
--tw-backdrop-contrast: ;
--tw-backdrop-grayscale: ;
--tw-backdrop-hue-rotate: ;
--tw-backdrop-invert: ;
--tw-backdrop-opacity: ;
--tw-backdrop-saturate: ;
--tw-backdrop-sepia: ;
--tw-contain-size: ;
--tw-contain-layout: ;
```

```
--tw-contain-paint: ;
--tw-contain-style: ;
}
.sr-only {
position: absolute;
width: 1px;
height: 1px;
padding: 0;
margin: -1px;
overflow: hidden;
clip: rect(0, 0, 0, 0);
white-space: nowrap;
border-width: 0;
}
.fixed {
position: fixed;
}
.absolute {
position: absolute;
}
.relative {
position: relative;
}
.inset-0 {
inset: 0px;
```

```
}
.inset-x-0 {
left: 0px;
right: 0px;
}
.inset-y-0 {
top: 0px;
bottom: 0px;
}
.-bottom-12 {
bottom: -3rem;
}
.-left-12 {
left: -3rem;
}
.-right-12 {
right: -3rem;
}
.-top-12 {
top: -3rem;
}
.bottom-0 {
bottom: 0px;
}
.left-0 {
```

```
left: 0px;
}
.left-1\/2 {
left: 50%;
}
.right-0 {
right: 0px;
}
.right-4 {
right: 1rem;
}
.top-0 {
top: 0px;
}
.top-1\/2 {
top: 50%;
}
.top-4 {
top: 1rem;
}
.z-50 {
z-index: 50;
}
.m-0 {
margin: 0px;
```

```
}
.mx-auto {
margin-left: auto;
margin-right: auto;
}
.-ml-4 {
margin-left: -1rem;
}
.-mt-4 {
margin-top: -1rem;
}
.ml-\[10px\] {
margin-left: 10px;
}
.mr-\[5px\] {
margin-right: 5px;
}
.mt-24 {
margin-top: 6rem;
}
.mt-4 {
margin-top: 1rem;
}
.mt-\[10px\] {
margin-top: 10px;
```

```
}
.mt-\[20px\] {
margin-top: 20px;
}
.mt-auto {
margin-top: auto;
}
.box-border {
box-sizing: border-box;
}
.flex {
display: flex;
}
.inline-flex {
display: inline-flex;
}
.grid {
display: grid;
}
.h-10 {
height: 2.5rem;
}
.h-11 {
height: 2.75rem;
}
```

```
.h-2 {
height: 0.5rem;
}
.h-4 {
height: 1rem;
}
.h-8 {
height: 2rem;
}
.h-9 {
height: 2.25rem;
}
.h-\[100\%\] {
height: 100%;
}
.h-\[200px\] {
height: 200px;
}
.h-\[20vh\] {
height: 20vh;
}
.h-\[300px\] {
height: 300px;
}
.h-\[70\%\] {
```

```
height: 70%;
}
.h-auto {
height: auto;
}
.h-full {
height: 100%;
}
.w-10 {
width: 2.5rem;
}
.w-3\/4 {
width: 75%;
}
.w-4 {
width: 1rem;
}
.w-8 {
width: 2rem;
}
.w-\[100\%\] {
width: 100%;
}
.w-\[100px\] {
width: 100px;
```

```
}
.w-\[150px\] {
width: 150px;
}
.w-\[32\%\] {
width: 32%;
}
.w-full {
width: 100%;
}
.min-w-0 {
min-width: 0px;
}
.min-w-\[200px\] {
min-width: 200px;
}
.shrink-0 {
flex-shrink: 0;
}
.grow-0 {
flex-grow: 0;
}
.basis-full {
flex-basis: 100%;
}
```

```
.-translate-x-1\/2 {
--tw-translate-x: -50%;
transform: translate(var(--tw-translate-x), var(--tw-translate-y))
rotate(var(--tw-rotate)) skewX(var(--tw-skew-x)) skewY(var(--tw-skew-y))
scaleX(var(--tw-scale-x)) scaleY(var(--tw-scale-y));
}
.-translate-y-1\/2 {
--tw-translate-y: -50%;
transform: translate(var(--tw-translate-x), var(--tw-translate-y))
rotate(var(--tw-rotate)) skewX(var(--tw-skew-x)) skewY(var(--tw-skew-y))
scaleX(var(--tw-scale-x)) scaleY(var(--tw-scale-y));
}
.rotate-90 {
--tw-rotate: 90deg;
transform: translate(var(--tw-translate-x), var(--tw-translate-y))
rotate(var(--tw-rotate)) skewX(var(--tw-skew-x)) skewY(var(--tw-skew-y))
scaleX(var(--tw-scale-x)) scaleY(var(--tw-scale-y));
}
.list-disc {
list-style-type: disc;
}
.flex-col {
flex-direction: column;
}
.flex-col-reverse {
```

```
flex-direction: column-reverse;
}
.flex-wrap {
flex-wrap: wrap;
}
.items-center {
align-items: center;
}
.justify-center {
justify-content: center;
}
.justify-between {
justify-content: space-between;
}
.gap-1 {
gap: 0.25rem;
}
.gap-1\.5 {
gap: 0.375rem;
}
.gap-2 {
gap: 0.5rem;
}
.gap-4 {
gap: 1rem;
```

```
}
.space-y-1 > :not([hidden]) ~ :not([hidden]) {
--tw-space-y-reverse: 0;
margin-top: calc(0.25rem * calc(1 - var(--tw-space-y-reverse)));
margin-bottom: calc(0.25rem * var(--tw-space-y-reverse));
}
.space-y-1\.5 > :not([hidden]) ~ :not([hidden]) {
--tw-space-y-reverse: 0;
margin-top: calc(0.375rem * calc(1 - var(--tw-space-y-reverse)));
margin-bottom: calc(0.375rem * var(--tw-space-y-reverse));
}
.space-y-2 > :not([hidden]) ~ :not([hidden]) {
--tw-space-y-reverse: 0;
margin-top: calc(0.5rem * calc(1 - var(--tw-space-y-reverse)));
margin-bottom: calc(0.5rem * var(--tw-space-y-reverse));
}
.overflow-hidden {
overflow: hidden;
}
.overflow-y-scroll {
overflow-y: scroll;
}
.whitespace-nowrap {
white-space: nowrap;
}
```

```
.rounded-\[3px\] {
border-radius: 3px;
}
.rounded-full {
border-radius: 9999px;
}
.rounded-lg {
border-radius: var(--radius);
}
.rounded-md {
border-radius: calc(var(--radius) - 2px);
}
.rounded-sm {
border-radius: calc(var(--radius) - 4px);
}
.rounded-t-\[10px\] {
border-top-left-radius: 10px;
border-top-right-radius: 10px;
}
.border {
border-width: 1px;
}
.border-b {
border-bottom-width: 1px;
}
```

```
.border-b-\[1px\] {
border-bottom-width: 1px;
}
.border-l {
border-left-width: 1px;
}
.border-r {
border-right-width: 1px;
}
.border-t {
border-top-width: 1px;
}
.border-input {
border-color: hsl(var(--input));
}
.border-b-black {
--tw-border-opacity: 1;
border-bottom-color: rgb(0 0 0 / var(--tw-border-opacity));
}
.bg-background {
background-color: hsl(var(--background));
}
.bg-black {
--tw-bg-opacity: 1;
background-color: rgb(0 0 0 / var(--tw-bg-opacity));
```

```
}
.bg-black\/80 {
background-color: rgb(0 0 0 / 0.8);
}
.bg-card {
background-color: hsl(var(--card));
}
.bg-destructive {
background-color: hsl(var(--destructive));
}
.bg-muted {
background-color: hsl(var(--muted));
}
.bg-primary {
background-color: hsl(var(--primary));
}
.bg-secondary {
background-color: hsl(var(--secondary));
}
.bg-slate-400 {
--tw-bg-opacity: 1;
background-color: rgb(148 163 184 / var(--tw-bg-opacity));
}
.bg-white {
--tw-bg-opacity: 1;
```

```
background-color: rgb(255 255 255 / var(--tw-bg-opacity));
}
.p-4 {
padding: 1rem;
}
.p-6 {
padding: 1.5rem;
}
.p-\[5px\] {
        padding: 5px;
```