TI	Dokumentacja Projektu JavaScript
Autor	Oliwer Strzałka
Kierunek, rok	Informatyka i ekonometria, II rok, st. Stacjonarne (3.5-l)
Temat	Projekt i implementacja mechanizmu łamania i justowania tekstu z podziałem wyrazów

Tematyka projektu

Głównym założeniem projektu jest stworzenie aplikacji webowej umożliwiającej zaawansowane łamanie i justowanie tekstu w języku polskim, zaimplementowanej w czystym JavaScript bez użycia zewnętrznych bibliotek. Aplikacja ma na celu prezentację autorskiego algorytmu, który automatycznie formatuje tekst według zadanych parametrów, uwzględniając specyfikę języka polskiego. Podstawową funkcjonalnością aplikacji powinna być możliwość wprowadzenia dowolnego tekstu przez użytkownika, konfiguracji parametrów przetwarzania takich jak szerokość linii, włączenie lub wyłączenie łamania wyrazów oraz justowania, a następnie wyświetlenie sformatowanego tekstu z zachowaniem podanych ustawień. Dodatkowo, aplikacja powinna dostarczać szczegółowych statystyk dotyczących przetworzonego tekstu, takich jak liczba słów, liczba wygenerowanych linii oraz liczba zastosowanych łamań wyrazów. Ważnym aspektem projektu jest zapewnienie responsywnego i intuicyjnego interfejsu użytkownika, który będzie działał poprawnie na różnych urządzeniach – od komputerów stacjonarnych po smartfony. Kolejnym istotnym założeniem jest implementacja dynamicznej aktualizacji wyników w czasie rzeczywistym w odpowiedzi na zmiany parametrów, co pozwoli użytkownikowi na bieżąco obserwować efekty modyfikacji ustawień. Ostatnim, niemniej ważnym elementem projektu, jest

zapewnienie wysokiej wydajności algorytmu, umożliwiającej przetwarzanie nawet długich tekstów bez zauważalnych opóźnień, co jest kluczowe dla komfortu użytkowania.

Realizacja projektu

W celu realizacji założeń projektu, zdecydowano się na wykorzystanie języka JavaScript do implementacji algorytmu łamania i justowania tekstu, natomiast do stworzenia interfejsu użytkownika aplikacji użyto HTML oraz CSS, co zapewniło przejrzystość i intuicyjność obsługi. Dzięki odpowiedniemu projektowi interfejsu użytkownik może łatwo zarządzać ustawieniami przetwarzania tekstu, korzystając z czytelnych formularzy, suwaków i przycisków. Proces przetwarzania tekstu realizowany jest za pomocą klasy TextJustifier, która zawiera logikę algorytmu, oraz obiektu App, który zarządza interakcją z interfejsem. Po wprowadzeniu tekstu i ustawieniu parametrów, użytkownik może kliknąć przycisk "Przetwórz", co uruchamia proces formatowania. Aby umożliwić użytkownikowi dynamiczną zmianę ustawień, zastosowano nasłuchiwanie zdarzeń na suwaku szerokości linii oraz przełącznikach, które automatycznie wywołują ponowne przetwarzanie przy każdej modyfikacji. Dla zapewnienia dodatkowej wygody, zaimplementowano przycisk resetowania, który przywraca domyślne wartości wszystkich ustawień. W celu poprawienia organizacji interfejsu, dodano dwa główne panele: panel ustawień z formularzem i kontrolkami oraz panel wyników z wyświetlaczem sformatowanego tekstu i statystykami, które są wyraźnie oddzielone wizualnie. Dodatkowo, aplikacja oferuje funkcję statystyk, które są obliczane na podstawie przetworzonego tekstu i prezentowane w formie liczbowej. Ważnym elementem aplikacji jest możliwość łamania długich wyrazów, które nie mieszczą się w linii, z użyciem łącznika zgodnie z zasadami języka polskiego. Justowanie tekstu realizowane jest przez równomierne rozłożenie odstępów między słowami w linii, tak aby linia była wyrównana do obu marginesów, z zachowaniem czytelności. Aplikacja została zoptymalizowana pod kątem wydajności, wykorzystując efektywne struktury danych i algorytmy, aby przetwarzanie nawet obszernych tekstów odbywało się bez zauważalnych opóźnień

Testowanie aplikacji

Proces testowania aplikacji webowej został zaplanowany w celu dokładnej weryfikacji jej funkcjonalności, dostępności oraz optymalizacji, co zapewni użytkownikom komfortowe i bezproblemowe korzystanie z serwisu. Testy obejmują kilka kluczowych obszarów. Pierwszym krokiem jest sprawdzenie dostępności wszystkich sekcji strony na różnych urządzeniach. Witryna będzie testowana pod kątem poprawnego wyświetlania na komputerach, tabletach i smartfonach, aby upewnić się, że wygląda i działa prawidłowo w każdej rozdzielczości ekranu. W tym celu zostaną wykorzystane

narzędzia developerskie przeglądarek oraz symulatory urządzeń, które pozwolą ocenić responsywność strony. Kolejnym etapem jest optymalizacja aplikacji. W ramach tych testów zostanie przeanalizowany czas ładowania witryny oraz jej wydajność podczas przetwarzania tekstu. Narzędzia takie jak Google PageSpeed Insights i Lighthouse umożliwią ocenę szybkości ładowania oraz wskażą obszary wymagające poprawy. Zostaną również zweryfikowane rozmiary plików graficznych, arkuszy CSS oraz skryptów JavaScript, aby upewnić się, że są odpowiednio skompresowane i zoptymalizowane, co przełoży się na lepsze działanie aplikacji. Niezwykle ważnym elementem testów jest walidacja kodu. Zostanie ona przeprowadzona z użyciem narzędzi takich jak W3C Validator, aby upewnić się, że kod HTML i CSS jest zgodny z obowiązującymi standardami. Ponadto, przy pomocy narzędzia Can I Use, zostanie przeanalizowana kompatybilność użytych znaczników i stylów z różnymi przeglądarkami, co zagwarantuje spójne działanie strony na takich platformach jak Edge, Opera i innych. Istotną częścią testów są również testy funkcjonalne, które mają na celu sprawdzenie poprawności działania interaktywnych elementów aplikacji. Weryfikacji poddane zostaną pole tekstowe, suwak szerokości linii, przełączniki łamania i justowania, przyciski przetwarzania i resetowania oraz pole wynikowe. Dodatkowo, aplikacja zostanie przetestowana pod kątem obsługi różnych przypadków brzegowych, takich jak puste pole tekstowe, ekstremalnie długie wyrazy czy specjalne znaki, aby upewnić się, że algorytm działa zgodnie z założeniami. Ostatnim, ale równie ważnym elementem testów, jest analiza dostępności aplikacji dla osób z niepełnosprawnościami. Zostanie oceniona czytelność interfejsu, w tym kontrast kolorów oraz struktura nagłówków, a także obsługa za pomocą czytników ekranowych, co pozwoli na dostosowanie aplikacji do potrzeb szerokiego grona odbiorców.

Testowanie aplikacji

Proces testowania aplikacji obejmował kilka kluczowych etapów, które miały na celu zapewnienie, że aplikacja działa zgodnie z założeniami projektowymi oraz spełnia oczekiwania użytkowników. Testy te skupiały się zarówno na funkcjonalności, wydajności, jak i poprawności kodu.

Lista testów przeprowadzonych w ramach projektu:

• **Łamanie wyrazów:** Testowano poprawność łamania długich wyrazów z użyciem łącznika. Sprawdzono, czy wyrazy są dzielone w odpowiednich miejscach zgodnie z zasadami języka polskiego i czy łącznik pojawia się poprawnie w wynikowym tekście.

• **Justowanie tekstu:** Weryfikowano, czy tekst jest równomiernie justowany do obu marginesów, z odpowiednim rozkładem odstępów między słowami, oraz czy justowanie jest poprawne zarówno dla krótkich, jak i długich linii.

• **Zmiana szerokości linii:** Testowano, czy zmiana szerokości linii za pomocą suwaka natychmiastowo wpływa na wynik formatowania i czy tekst jest poprawnie dopasowywany do nowej szerokości bez utraty treści.

• **Przełączanie funkcji łamania i justowania:** Sprawdzono, czy przełączniki poprawnie włączają i wyłączają odpowiednie funkcje, oraz czy zmiana stanu przełączników jest natychmiast odzwierciedlana w wyniku.

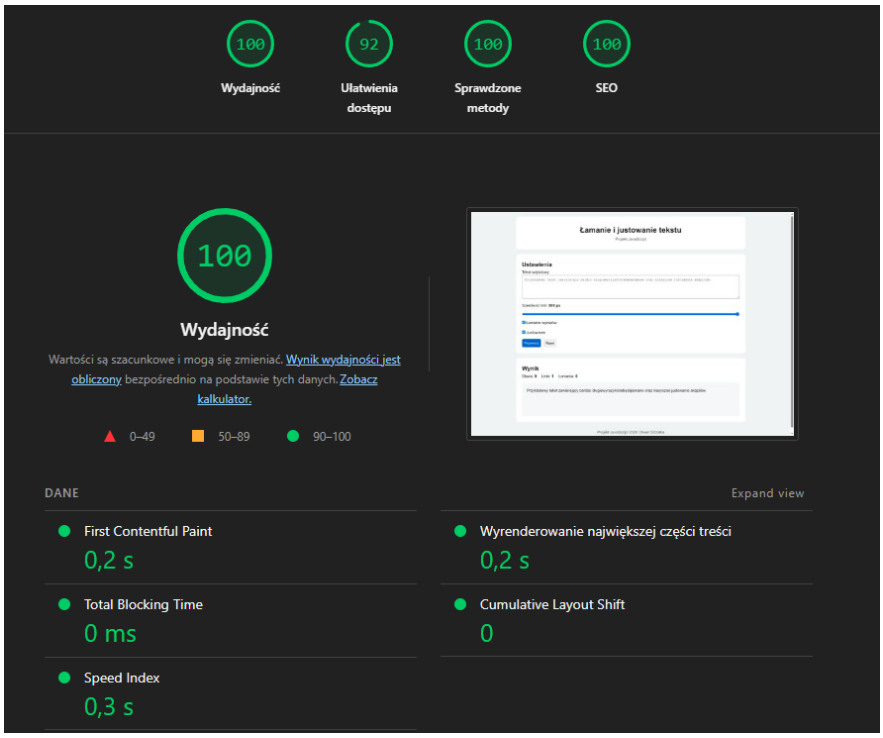
• **Obliczanie statystyk:** Testowano, czy statystyki (liczba słów, linii, łamań) są poprawnie obliczane na podstawie przetworzonego tekstu i czy są dokładnie wyświetlane w interfejsie.

• **Resetowanie ustawień:** Weryfikowano, czy przycisk reset przywraca domyślne ustawienia (szerokość 500px, włączone łamanie i justowanie) i czy tekst jest przetwarzany ponownie z tymi ustawieniami.

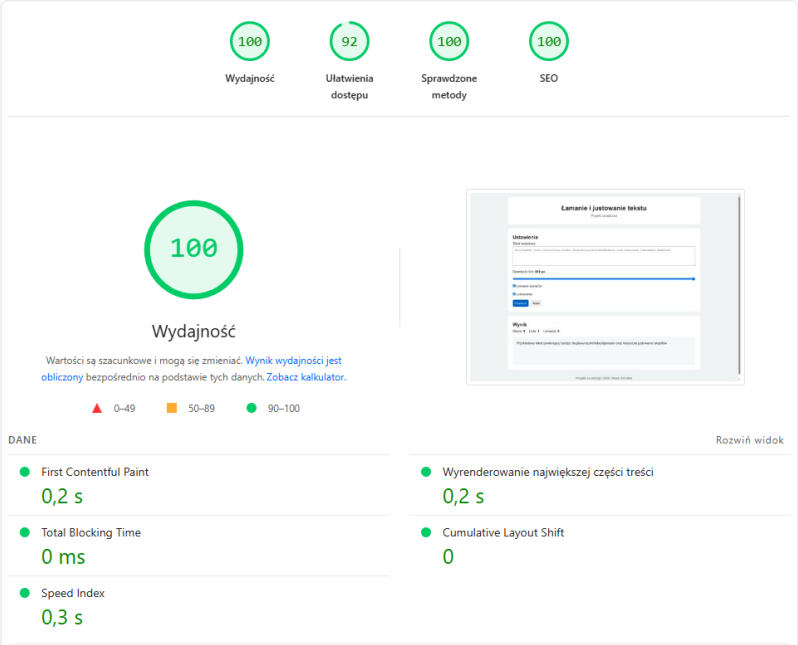
• **Responsywność interfejsu:** Sprawdzono, czy interfejs dostosowuje się do różnych rozdzielczości ekranu, czy elementy zachowują właściwe proporcje i czy aplikacja pozostaje funkcjonalna na urządzeniach mobilnych.

• **Obsługa długich i złożonych tekstów:** Testowano działanie aplikacji z bardzo długimi tekstami, tekstami zawierającymi znaki specjalne, interpunkcję i wielowierszowe akapity, aby upewnić się, że wydajność i poprawność są zachowane.

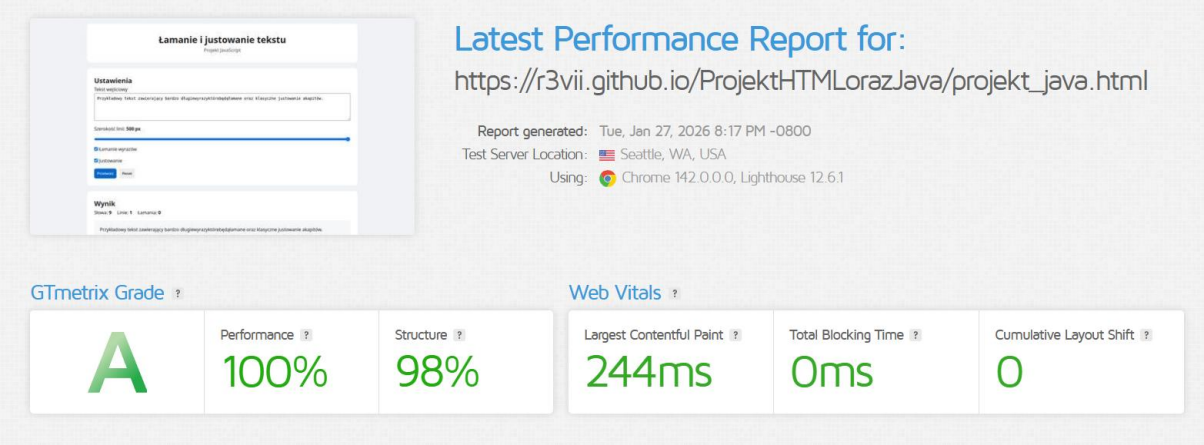
Testy optymalizacji



Rysunek 1. Wynik testu z narzędzia lighthouse



Rysunek 2. Wynik testu z narzędzia pagespeed.web

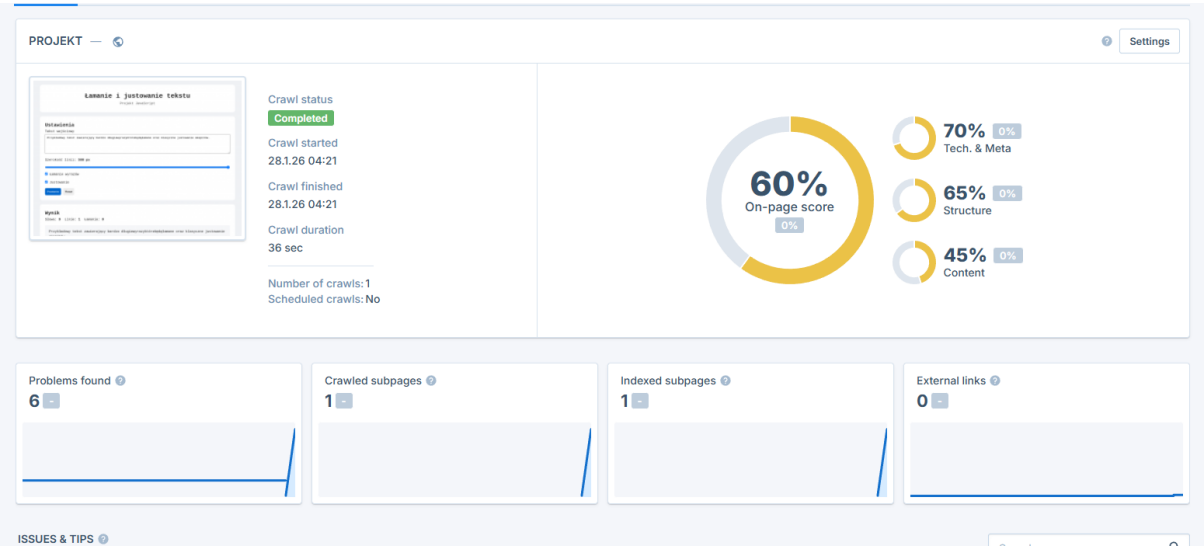


Rysunek 3. Rysunek z witryny GTmetrix

Wyniki testu optymalizacji wykazują, że aplikacja internetowa jest w dużej mierze zoptymalizowana pod względem wydajności, z uwzględnieniem kluczowych aspektów takich jak czas ładowania, minimalizacja zasobów oraz optymalizacja przetwarzania tekstu. Zauważalnym punktem jest całkowity brak zewnętrznych bibliotek, co pozytywnie wpływa na czas ładowania strony i jej ogólną wydajność. Aplikacja ładuje się szybko nawet przy słabszych połączeniach internetowych, co zapewnia komfort użytkowania. Dodatkowo, testy sugerują, że aplikacja mogłaby zyskać na drobnych optymalizacjach pod kątem dostępności. Chociaż większość elementów jest zgodna z podstawowymi wymaganiami dostępności, istnieje przestrzeń do poprawy w zakresie wsparcia dla osób z niepełnosprawnościami, takich jak lepsza obsługa czytników

ekranu czy poprawa kontrastu wizualnego w wybranych sekcjach. Implementacja tych usprawnień poprawiłaby ogólną dostępność aplikacji, czyniąc ją bardziej dostosowaną do potrzeb szerszej grupy użytkowników.

Raport SEO



Rysunek 5. Rysunek testu z witryny seobility.net

Test wyszedł pozytywnie, brakuje jednak samego www redirect przez co test mógł zostać zaniżony. Dodatkowo pojawiają sięminimalne błędy które mogą wpływać na taki wynik, przykładem jest jeden plik który ma lekko ponad 500 linijek kodu i jest brany jako ostrzeżenie przez ten test

Implementacja

Pełna dokumentacja techniczna kodu została wygenerowana przy użyciu narzędzia JSDoc i znajduje się w katalogu docs/ projektu. Dokumentacja zawiera:

- **Szczegółowy opis wszystkich klas i metod**
- **Typy parametrów i wartości zwracanych**
- **Przykłady użycia funkcji**
- **Struktury danych i zależności**

Aby przeglądać dokumentację:

1. Otwórz plik docs/index.html w dowolnej przeglądarce internetowej
2. Możesz nawigować po:

a. Klasie TextJustifier - głównym silniku justowania tekstu
b. Obiekcie App - zarządzającym interfejsem użytkownika
c. Wszystkich metodach publicznych i prywatnych

Dokumentacja została wygenerowana automatycznie na podstawie komentarzy w kodzie źródłowym, co zapewnia jej aktualność i spójność z implementacją

Przetwarzanie tekstu

Zaimplementowano funkcję process(), która pobiera dane z formularza, tworzy obiekt z opcjami, a następnie przekazuje go do metody justify() klasy TextJustifier. Gdy użytkownik wprowadzi tekst w polu tekstowym i skonfiguruje parametry (szerokość linii, łamanie, justowanie), po kliknięciu przycisku "Przetwórz" funkcja process() zostaje uruchomiona. Funkcja odczytuje wartości wpisane przez użytkownika w odpowiednich polach formularza HTML za pomocą metody document.getElementById. Przykładowo:

• Tekst wejściowy (text-input)
• Szerokość linii (width-slider)
• Łamanie wyrazów (hyphenate-toggle)
• Justowanie (justify-toggle)

Zanim tekst zostanie przetworzony, funkcja może sprawdzić, czy użytkownik wprowadził tekst. Jeśli pole tekstowe jest puste, aplikacja wyświetla standardowy wynik z komunikatem informacyjnym.

Po pomyślnym pobraniu danych funkcja tworzy obiekt opcji, który zawiera:

• width: szerokość linii w pikselach,
• justify: wartość logiczna wskazująca, czy justować tekst,
• hyphenate: wartość logiczna wskazująca, czy łamać wyrazy.

Obiekt opcji jest następnie przekazywany do metody justify obiektu justifier (instancji TextJustifier) wraz z tekstem. Metoda justify zwraca sformatowany tekst w formie HTML, gdzie każda linia jest zawarta w odpowiednim kontenerze z właściwym stylem justowania, a łamane wyrazy zawierają łącznik w miejscu podziału.

Po przetworzeniu tekstu funkcja wywołuje metodę getStats(), która oblicza statystyki (liczba słów, linii, łamań) i aktualizuje interfejs użytkownika, wyświetlając te wartości w przeznaczonych do tego polach.

```
    process() {
        const result = this.justifier.justify(this.text.value, {
            width: Number(this.widthSlider.value),
            justify: this.justify.checked,
            hyphenate: this.hyphen.checked
        });

        this.output.innerHTML = result;

        const stats = this.justifier.getStats(result);
        this.wordCount.textContent = stats.words;
        this.lineCount.textContent = stats.lines;
        this.hyphenCount.textContent = stats.hyphens;
    }
};

document.addEventListener('DOMContentLoaded', () => App.init());
```

Rysunek 6. Kod z implementacją funkcji process i przykładem jej wywołania

Źródła

- Technologie i biblioteki: HTML5, CSS3, JavaScript (ES6+)
- Narzędzia programistyczne: Visual Studio Code, Git, Chrome DevTools
- **Licencje i prawa autorskie (do fragmentów tekstu/zdjęć)**

Prompty AI

- Czym jest justowanie i jak może wyglądać przykładowa implementacja
- Błąd “+ CategoryInfo : SecurityError: (:) [], PSSecurityException + FullyQualifiedErrorId : UnauthorizedAccess” podczas instalacji jsdoc i jak go naprawić?