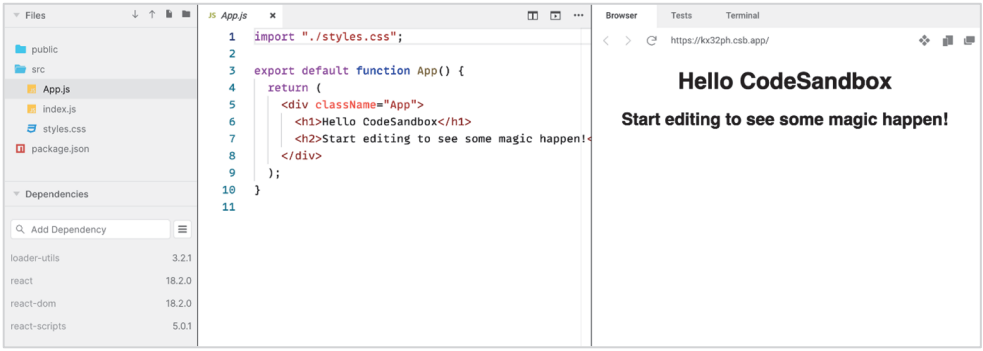
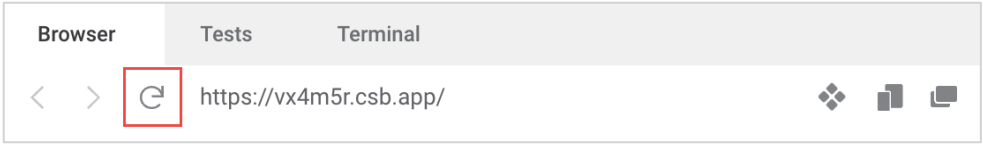
Kolorowe wersje rysunków w książce „React i TypeScript. Reaktywne tworzenie stron internetowych dla początkujących. Wydanie II”

Rozdział 1. Wprowadzenie do Reacta



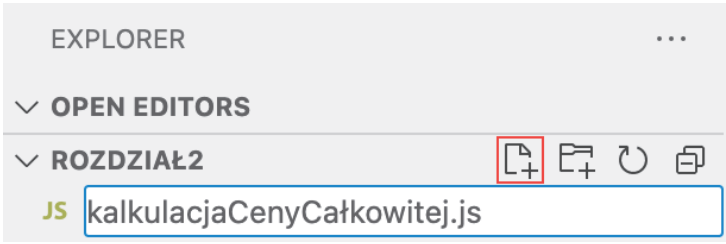
Rysunek 1.1. Projekt Reacta w CodeSandboxie

Rysunek 1.5. Przycisk close w komponencie Alert

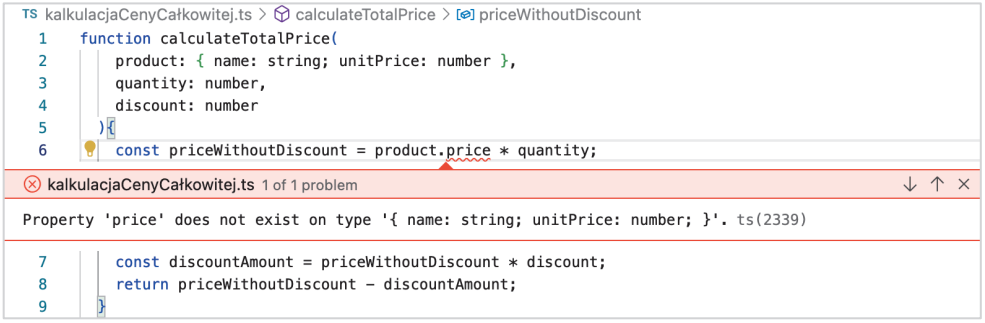


Rysunek 1.7. Opcja odświeżania w panelu Browser

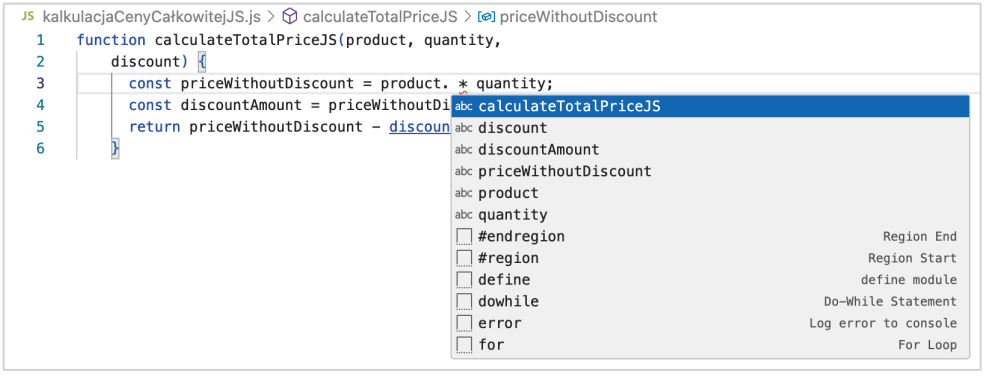
Rozdział 2. Wprowadzenie do TypeScriptu



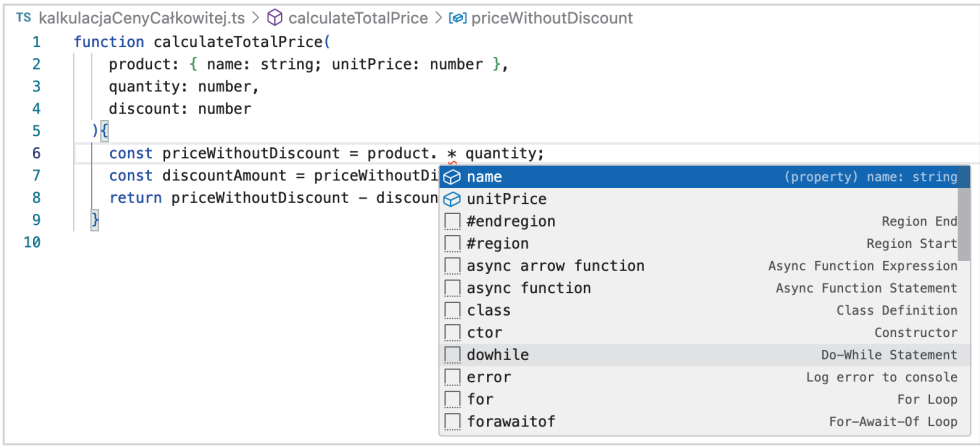
Rysunek 2.1. Tworzenie nowego pliku w Visual Studio Code



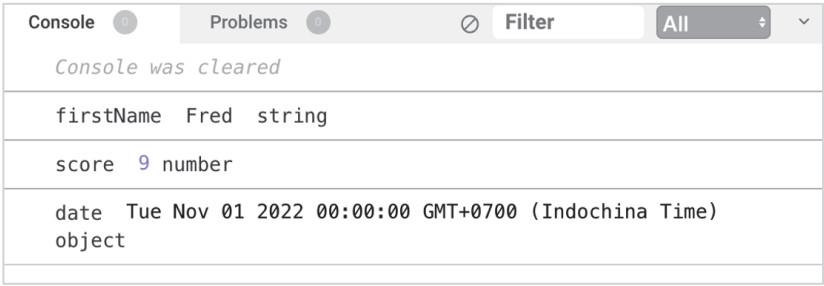
Rysunek 2.2. Podświetlony błąd typowania



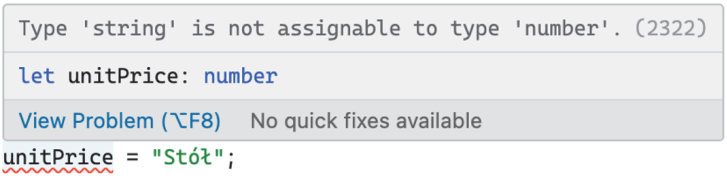
Rysunek 2.3. IntelliSense w pliku JavaScriptu



Rysunek 2.4. IntelliSense w pliku TypeScriptu



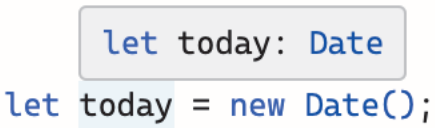
Rysunek 2.5. Przykłady typów w JavaScripcie



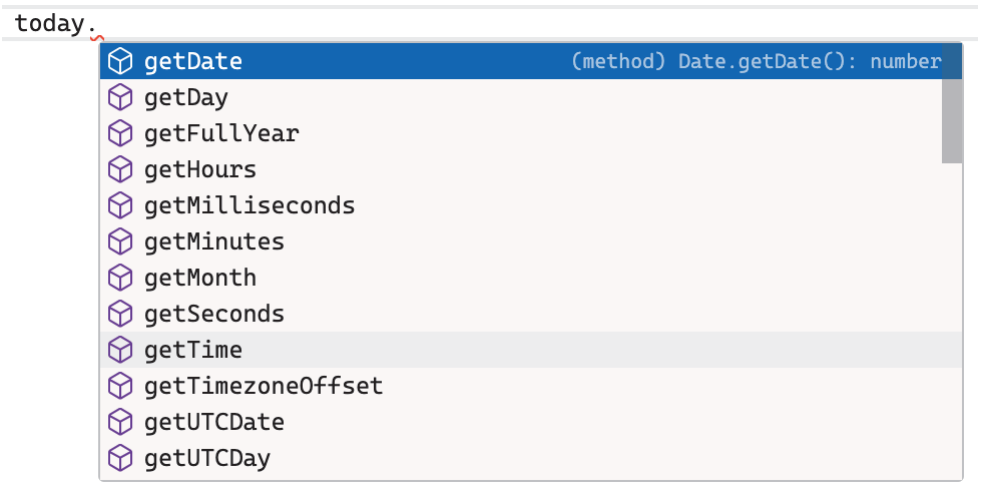
Rysunek 2.7. Wykrycie błędu typowania

Rysunek 2.8. Dwa wykryte błędy typów

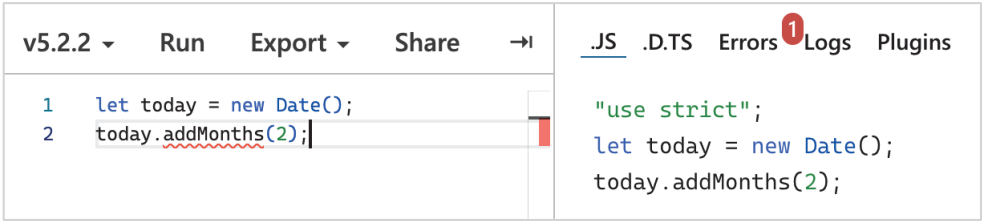
Rysunek 2.9. Najechanie kursorem na zmienną pokazuje jej typ



Rysunek 2.10. Dymek potwierdzający, że today ma typ Date



Rysunek 2.11. IntelliSense świetnie radzi sobie z datami

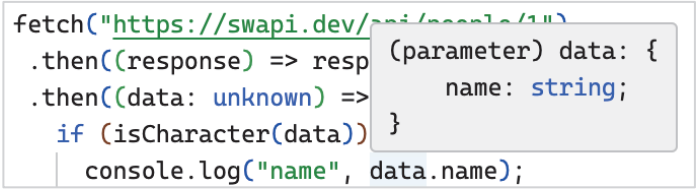


Rysunek 2.12. Błąd typu przy próbie użycia nieistniejącej funkcji z datą

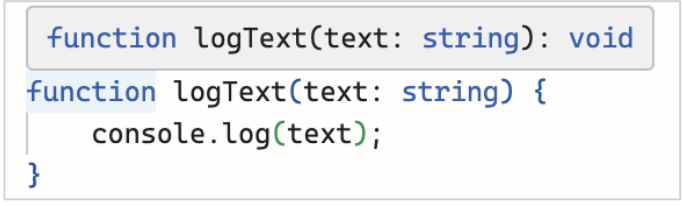
Rysunek 2.13. Zmienna otrzymała typ any

Rysunek 2.14. Atrybut firstName jest niezdefiniowany

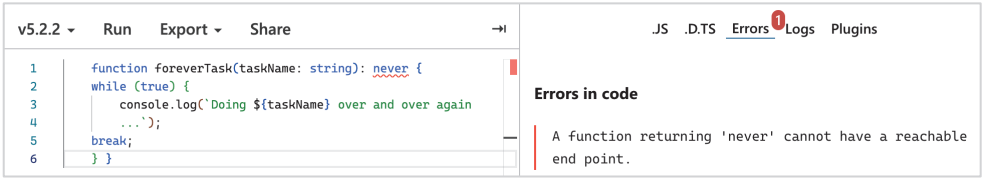
Rysunek 2.15. Błąd typowania dla parametru data oznaczonego jako unknown



Rysunek 2.16. Rozszerzenie typu zmiennej data



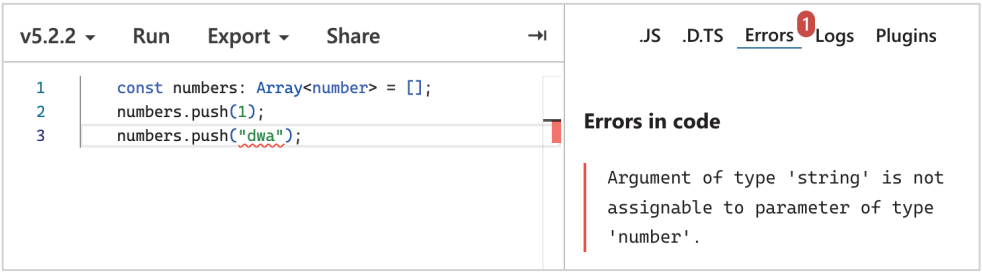
Rysunek 2.17. Typ void, który zwróci funkcja



Rysunek 2.18. Problem związany z użyciem typu never

```
function foreverTask(taskName: string): void
function foreverTask(taskName: string) {
while (true) {
    console.log(`Doing ${taskName} over and over again
...`);
} }
```

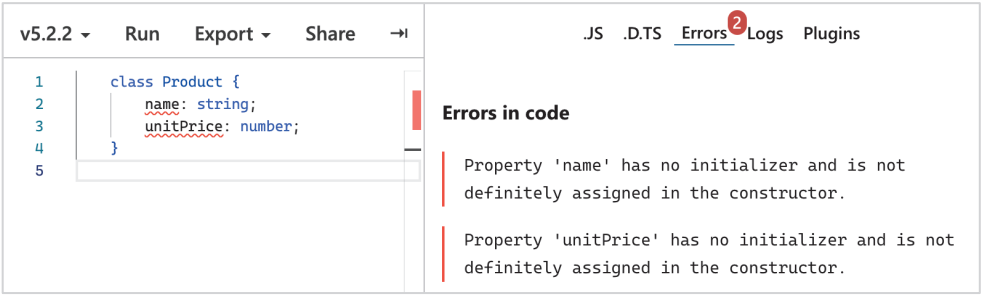
Rysunek 2.19. Samodzielnie wydedukowany typ jako void



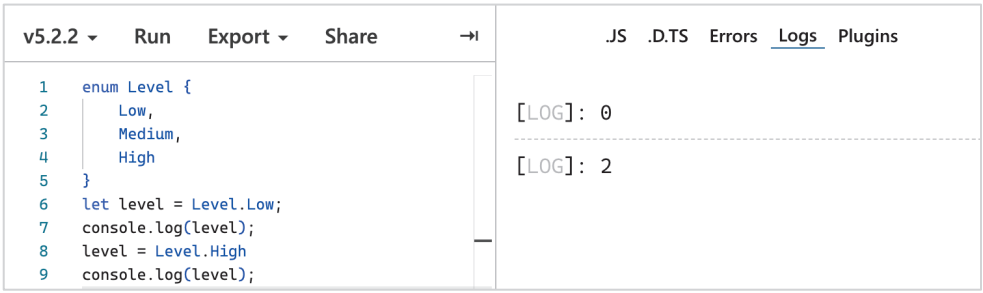
Rysunek 2.20. Dodanie tekstu do tablicy liczb wywołuje błąd typowania

```
const numbers: number[]
const numbers = [1, 2, 3];
```

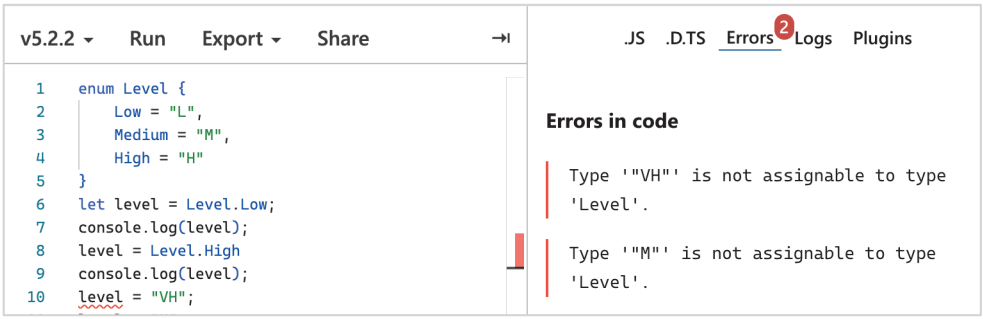
Rysunek 2.21. Wnioskowanie typu array w TypeScripcie



Rysunek 2.22. Problemy z typami w atrybutach klas



Rysunek. 2.23. Wartości typu wyliczeniowego



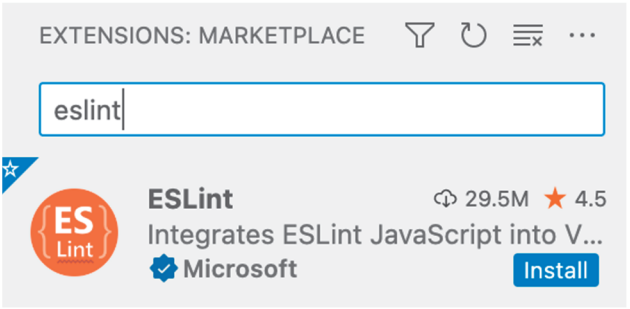
Rysunek 2.24. Dowód na to, że enumeracje oparte na ciągach znaków są bezpieczne pod względem typu

Rysunek 2.25. Błąd wynikający z niezgodności typu w unii

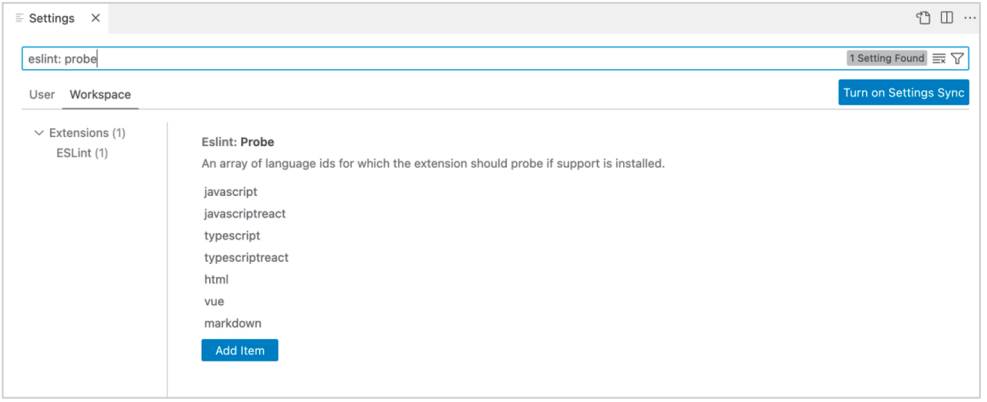
Rozdział 3. Konfiguracja Reacta i TypeScriptu

Rysunek 3.1. Aplikacja włączona w przeglądarce w trybie dla deweloperów

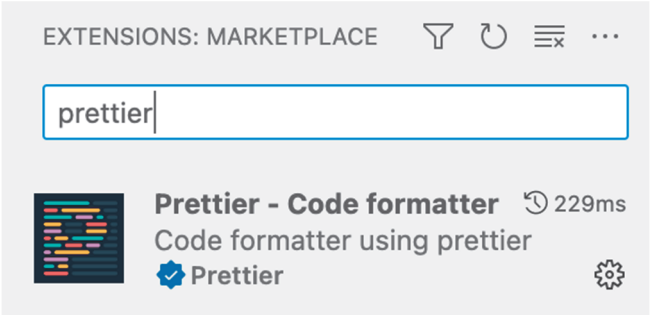
Rysunek 3.2. Automatyczne odświeżenie aplikacji



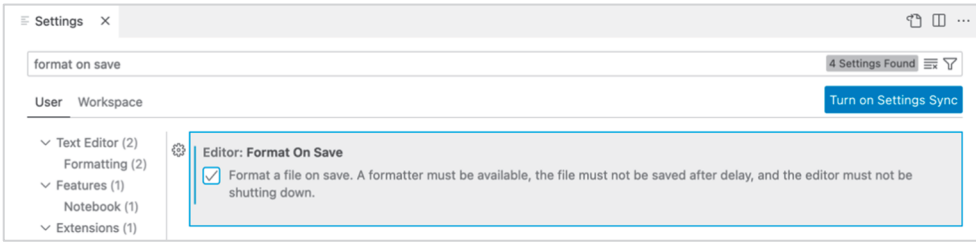
Rysunek 3.3. Rozszerzenie ESLint w Visual Studio Code



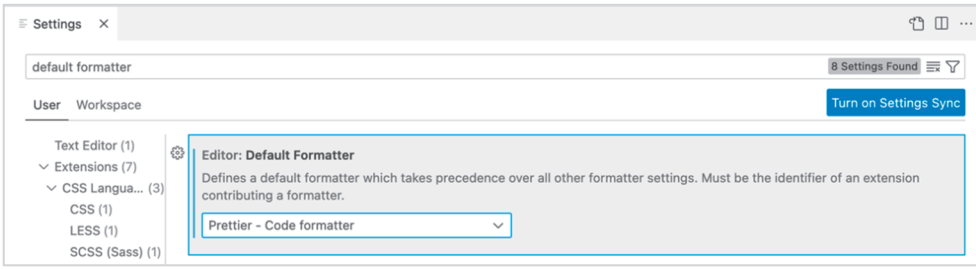
Rysunek 3.4. Konfiguracja ESLint Probe w Visual Studio Code



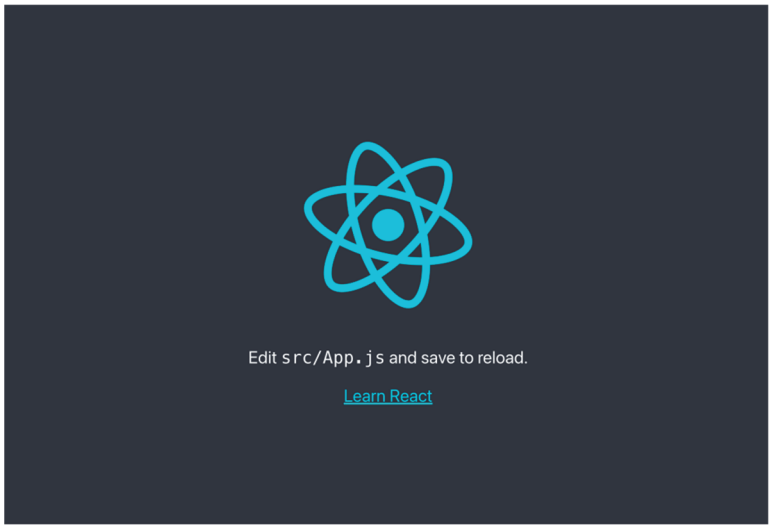
Rysunek 3.5. Rozszerzenie Prettier dla Visual Studio Code



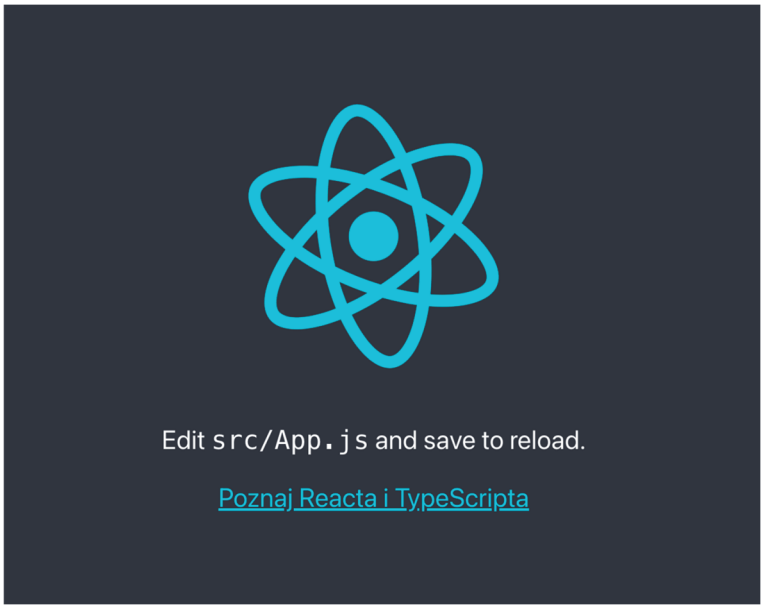
Rysunek 3.6. Ustawienie Format On Save w Visual Studio Code



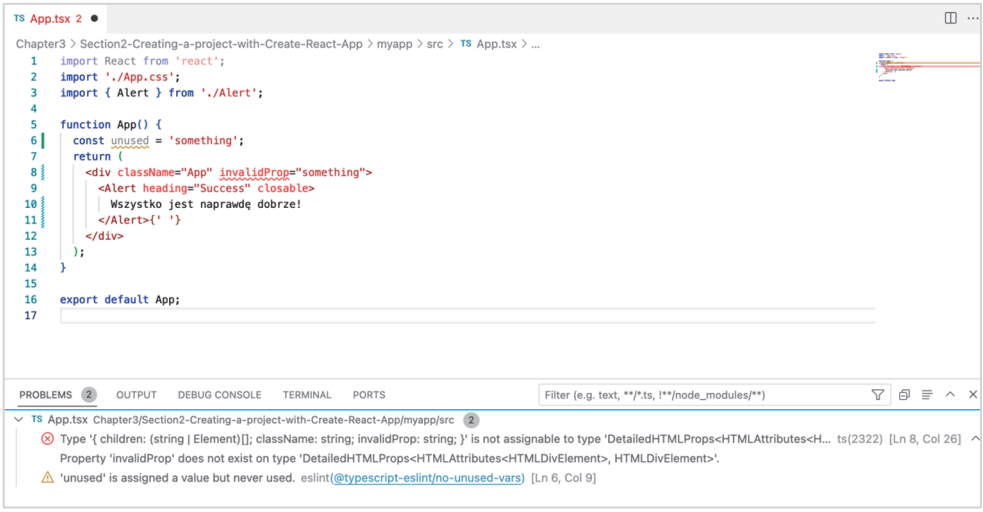
Rysunek 3.7. Wybór Prettiera jako domyślnego formatera w Visual Studio Code



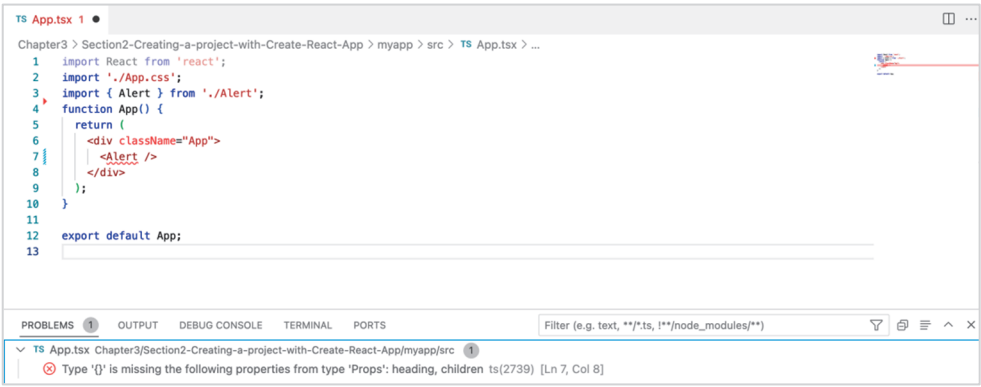
Rysunek 3.8. Działająca aplikacja Reacta w trybie dla programistów



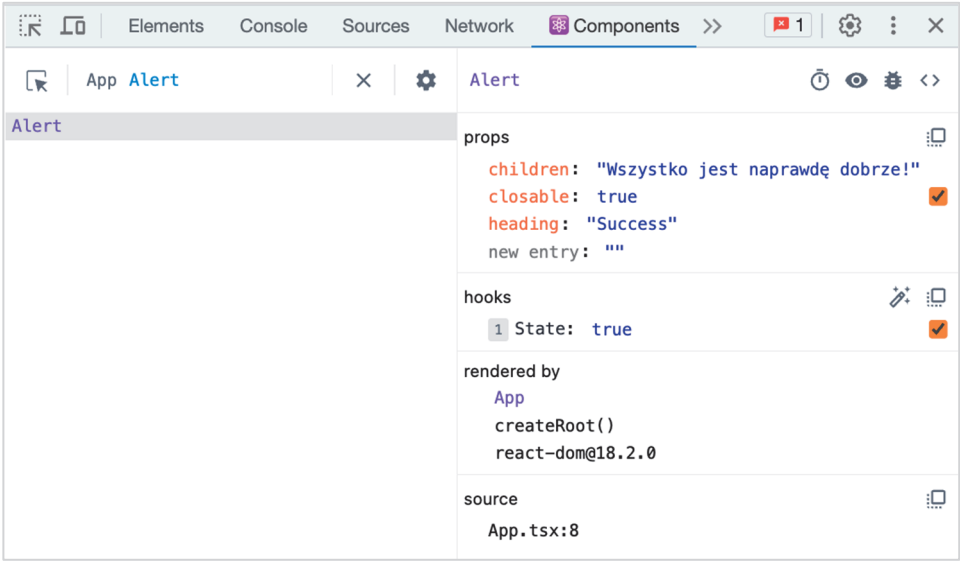
Rysunek 3.9. Odświeżony wygląd aplikacji Reacta



Rysunek 3.10. Nieprawidłowości wykryte przez Visual Studio Code

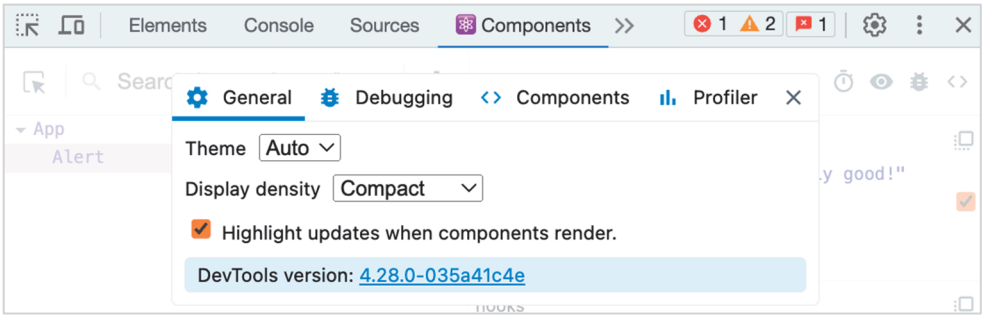


Rysunek 3.11. Błąd w komponencie Alert

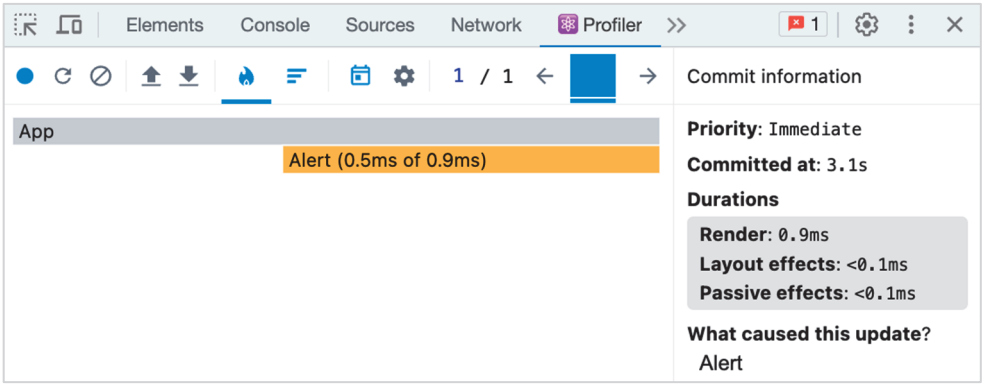


Rysunek 3.12. Panel Components w narzędziu React DevTools

Rysunek 3.13. Dokładna nazwa stanu po naciśnięciu różdżki

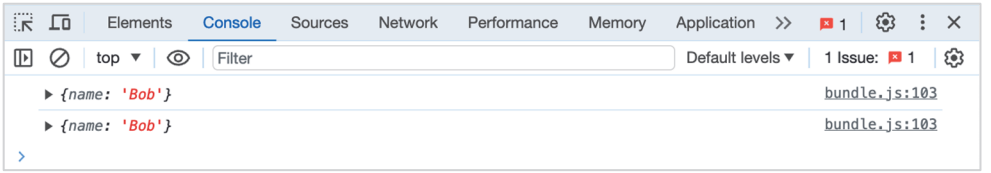


Rysunek 3.14. Zaznaczenie opcji, która zaznacza zmienione fragmenty po ponownym załadowaniu



Rysunek 3.16. Narzędzia React DevTools

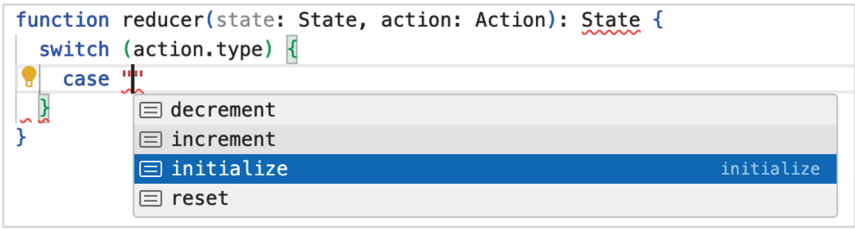
Rozdział 4. Korzystanie z hooków Reacta



Rysunek 4.1. Rezultat użycia efektu

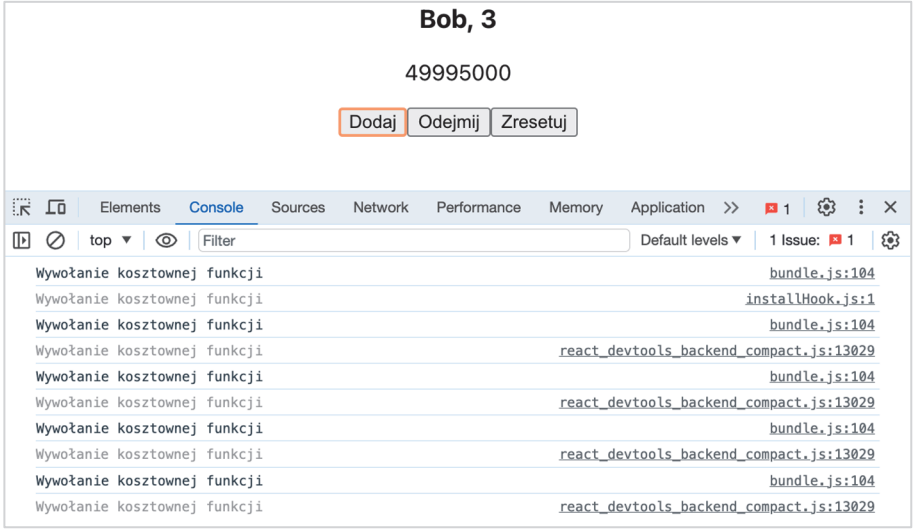
```
▶ Warning: useEffect must not return anything besides a      react-dom.development.js:86
function, which is used for clean-up.

It looks like you wrote useEffect(async () => ...) or returned a Promise. Instead,
write the async function inside your effect and call it immediately:

useEffect(() => {
  async function fetchData() {
    // You can await here
    const response = await MyAPI.getData(someId);
    // ...
  }
  fetchData();
}, [someId]); // Or [] if effect doesn't need props or state
```

Rysunek 4.2. Błąd związany z asynchronicznością efektu

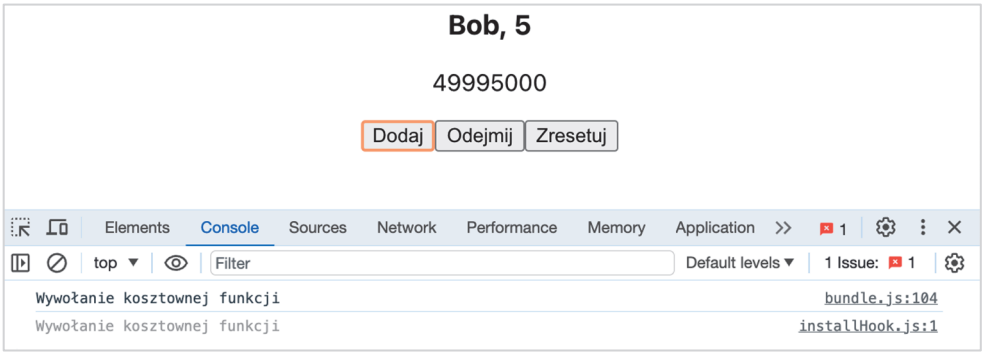


Rysunek 4.5. Podpowiedź IntelliSense w funkcji redukującej

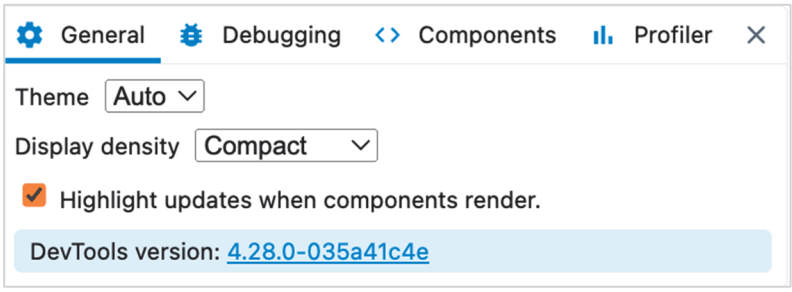
Rysunek 4.6. Zaznaczenie przycisku Dodaj



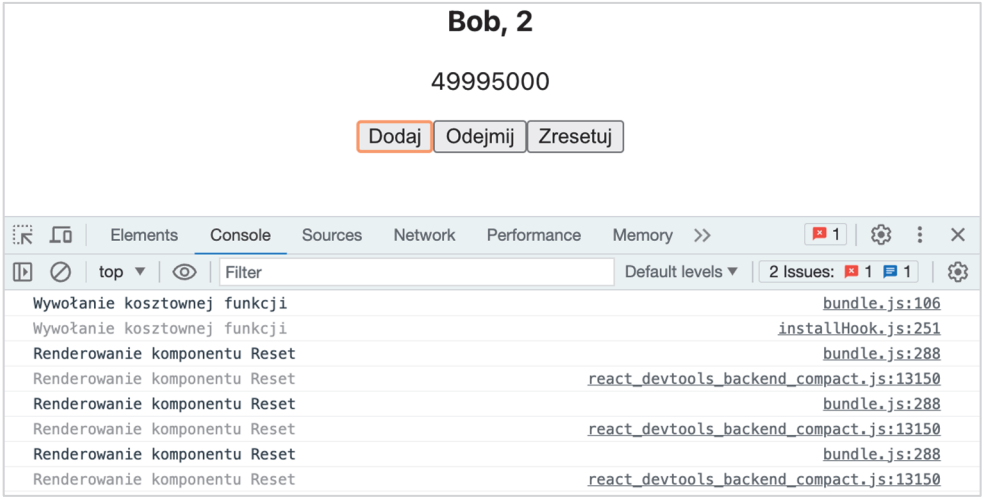
Rysunek 4.7. Wielokrotne wywołanie funkcji



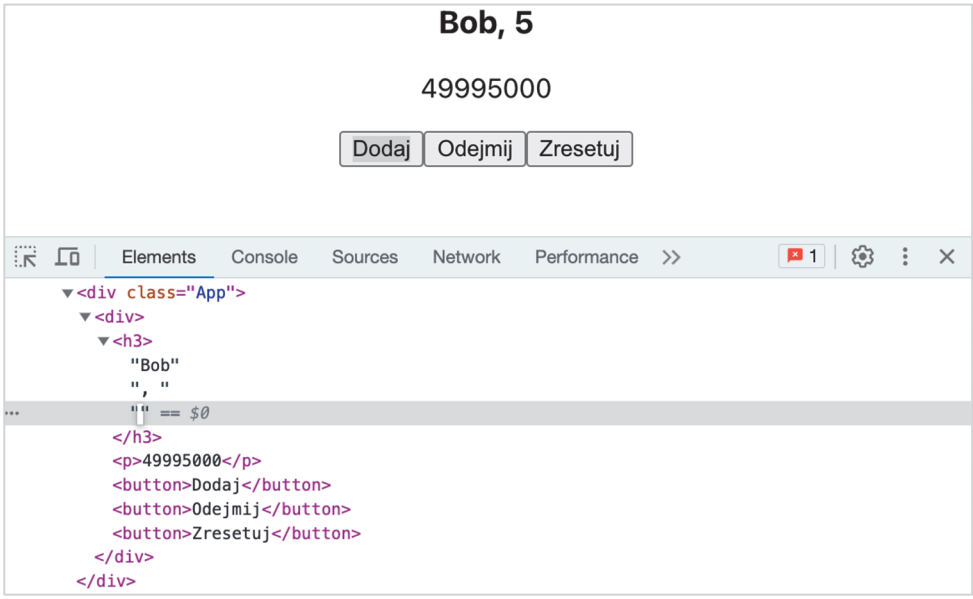
Rysunek 4.8. Zwrócenie zapamiętanych wartości funkcji



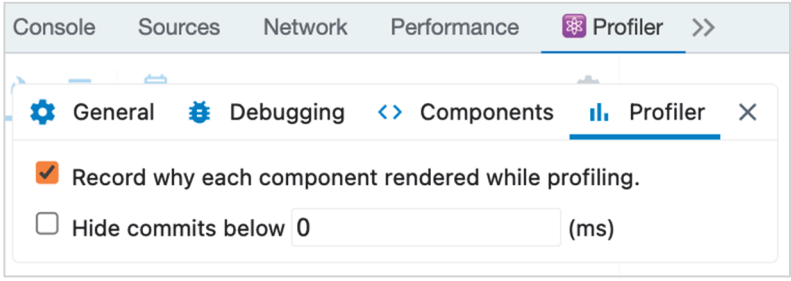
Rysunek 4.9. Opcja podświetlenia przy ponownym renderowaniu



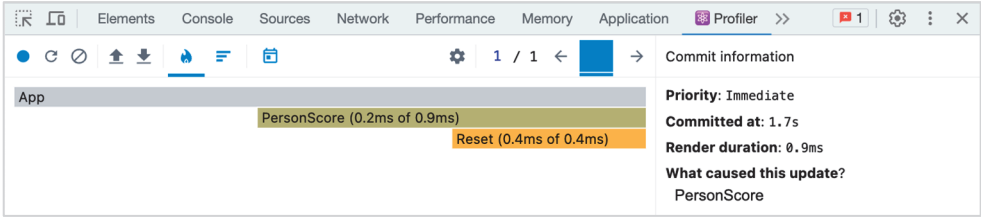
Rysunek 4.10. Zbędne ponowne renderowanie komponentu Reset



Rysunek 4.11. Zaktualizowany element h3 po renderowaniu



Rysunek 4.12. Upewnij się, że zaznaczona jest opcja Record why each component rendered while profiling

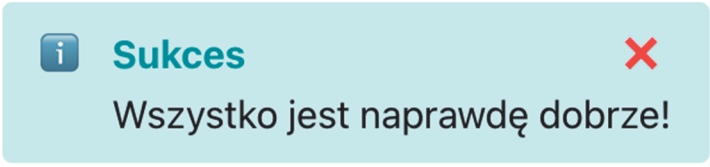


Rysunek 4.13. Szczegóły na temat ponownego renderowania komponentu Reset

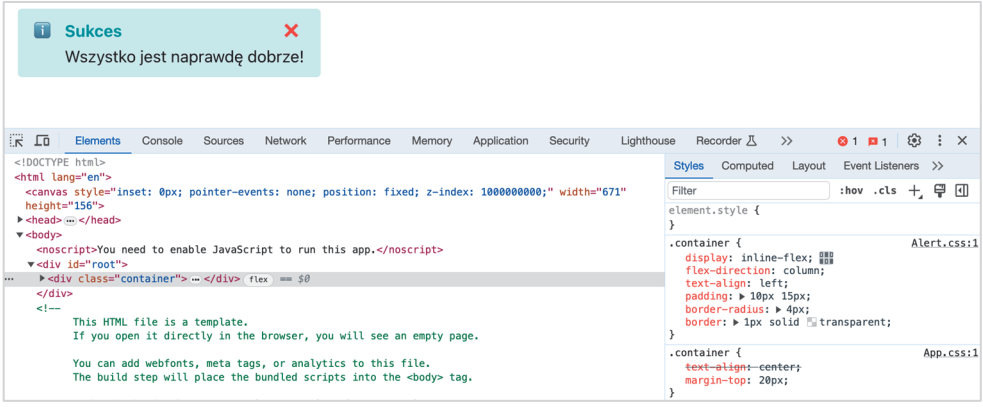
Rozdział 5. Stylizacja interfejsów w Reaccie

Rysunek 5.1. Element link w pliku index.html

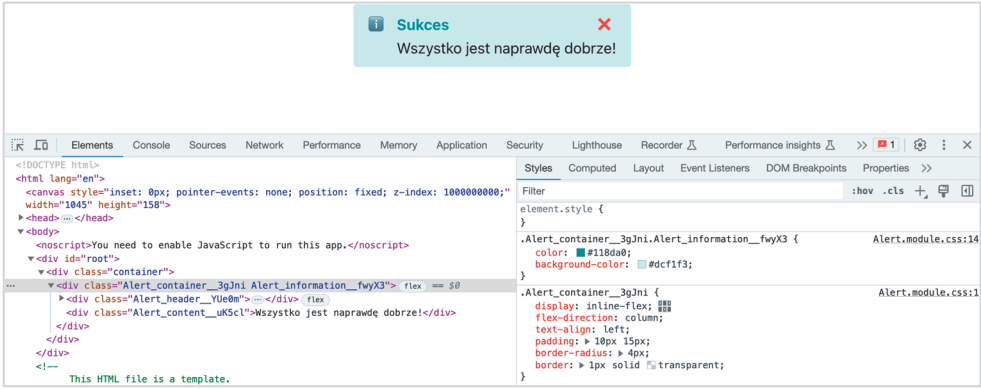
Rysunek 5.2. Dołączony plik CSS, wraz ze zbędną klasą App-header



Rysunek 5.3. Komponent Alert ostylowany przy użyciu standardowego CSS

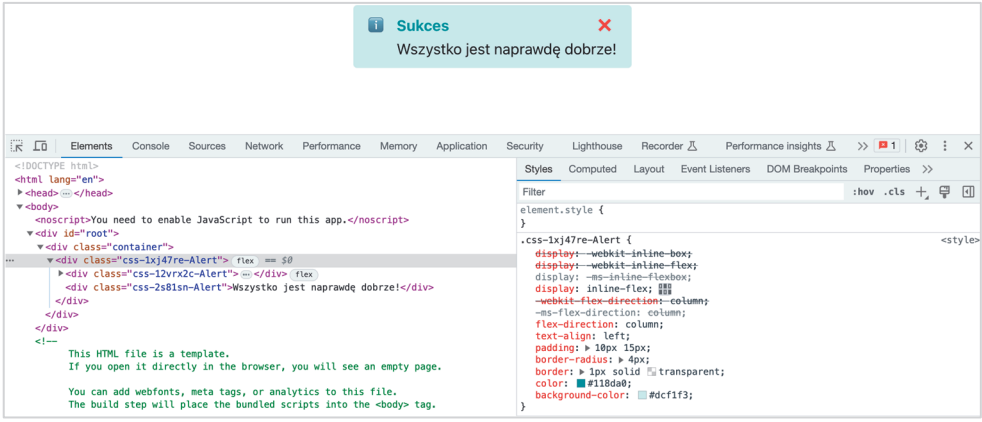


Rysunek 5.4. Kolizja klas CSS

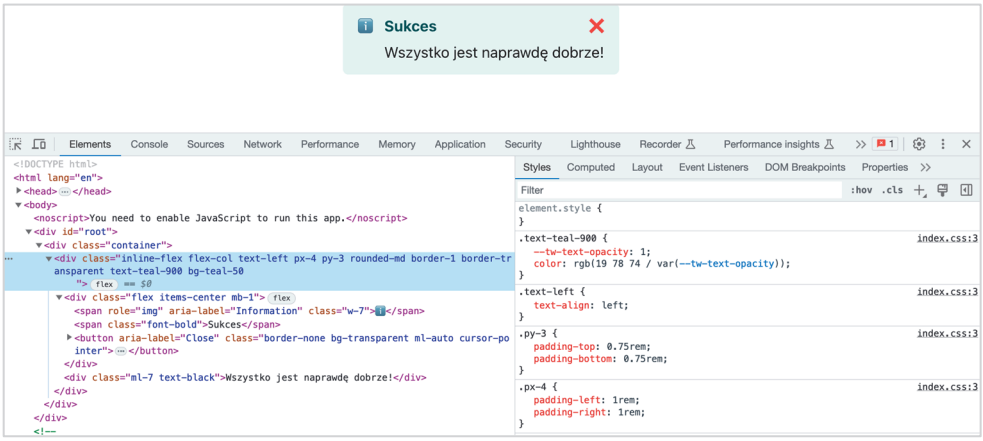


Rysunek 5.5. Nazwy klas w zakresie modułu CSS

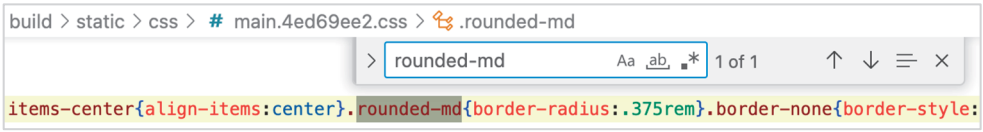
Rysunek 5.6. Niepotrzebna klasa CSS w zestawie CSS



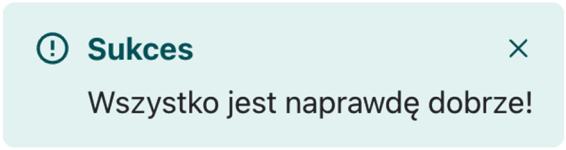
Rysunek 5.7. Nazwy klas Emotiona z podziałem na zakresy



Rysunek 5.8. Alert z zastosowanymi stylami z Tailwinda



Rysunek 5.9. Klasy CSS Tailwinda w złączonym pliku CSS

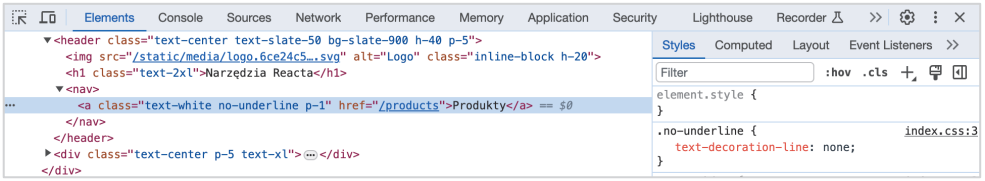


Rysunek 5.10. Alert z ikoną SVG

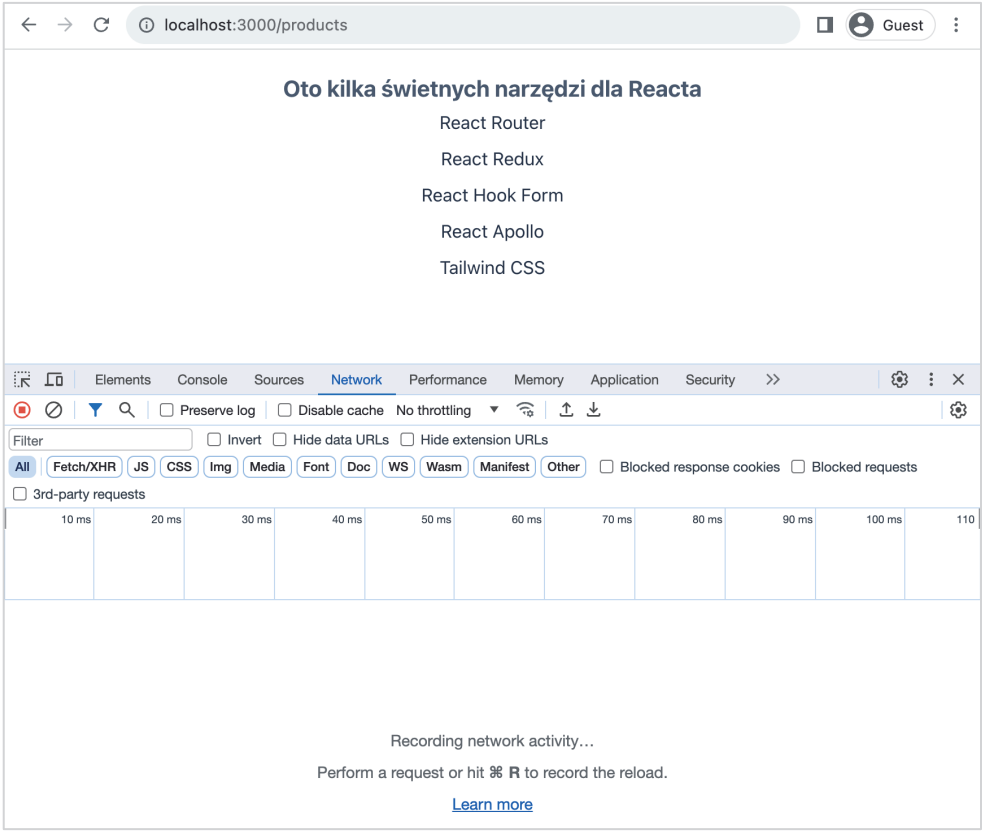
Rozdział 6. Routing przy użyciu biblioteki React Router

Rysunek 6.1. Standardowa strona biblioteki React Router pokazująca błąd

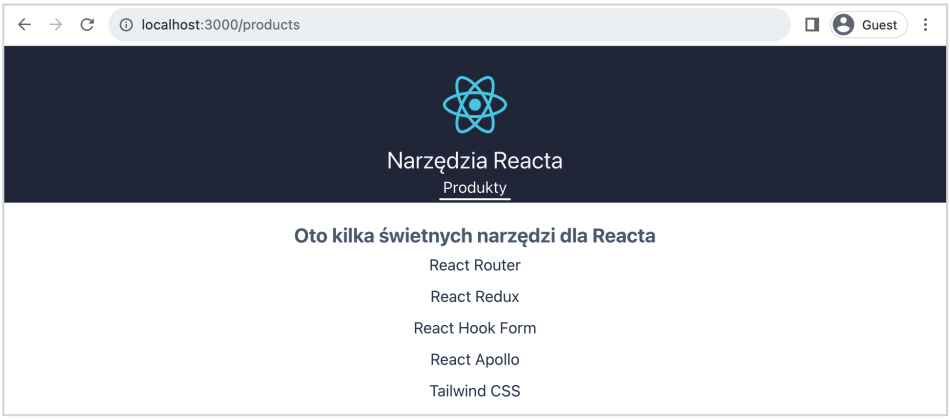
Rysunek 6.3. Główny nagłówek aplikacji



Rysunek 6.4. Analiza komponentu Header

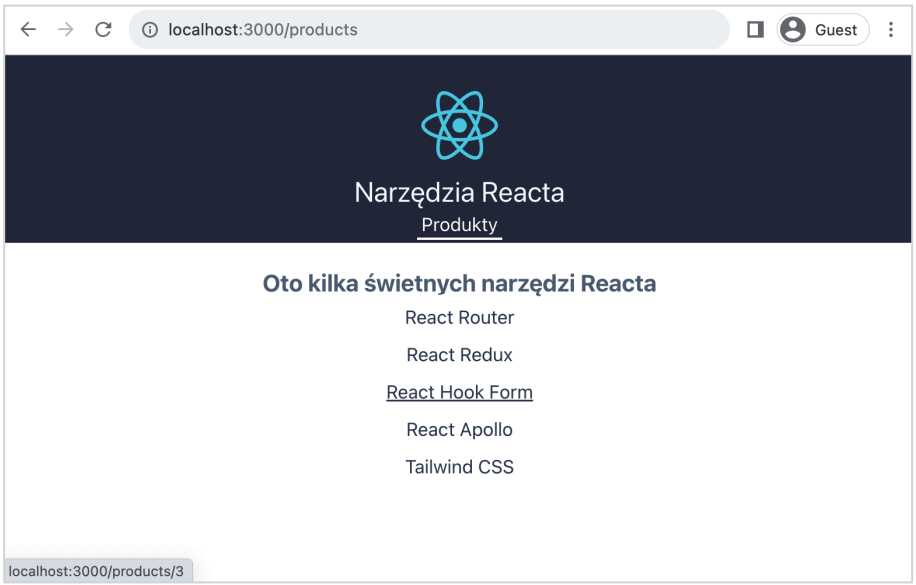


Rysunek 6.5. Nawigacja wewnętrzna aplikacji

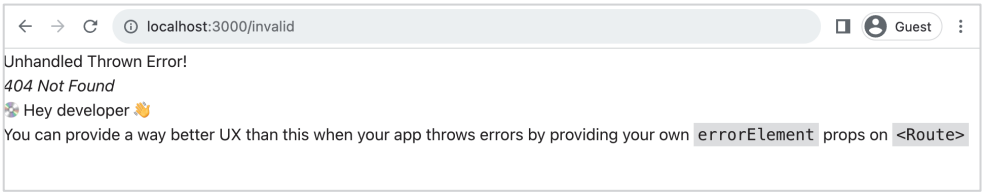


Rysunek 6.7. Główny nagłówek na stronie z listą produktów

Rysunek 6.8. Strona prezentująca produkt

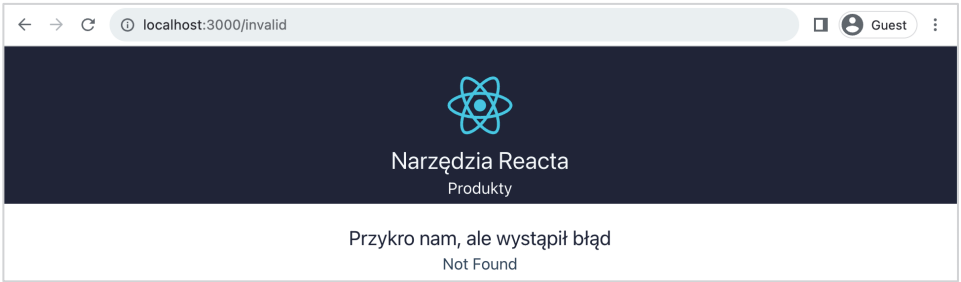


Rysunek 6.9. Linkowane produkty na liście



Rysunek 6.10. Domyślna strona błędu w bibliotece React Router

Rysunek 6.11. Nasza strona błędu

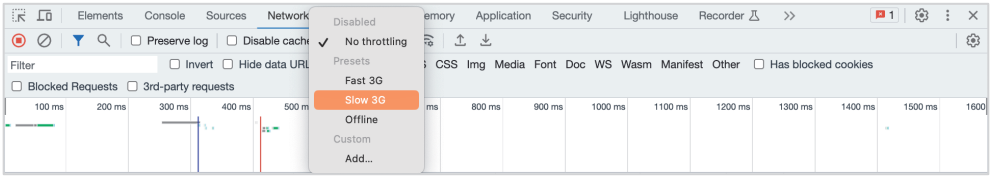


Rysunek 6.12. Strona błędu z informacją o wystąpieniu błędu

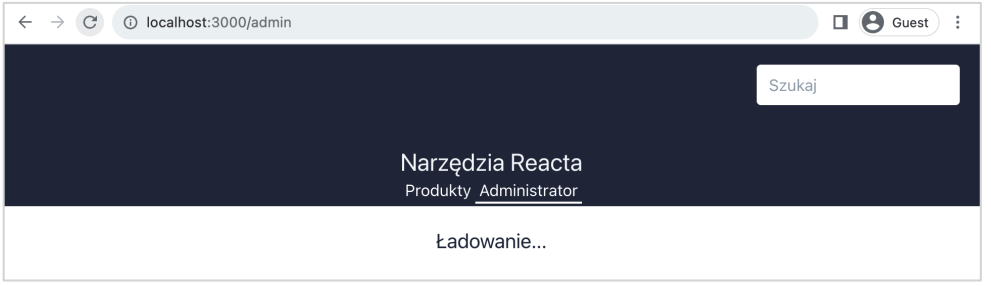
Rysunek 6.13. Strona powitalna

Rysunek 6.14. Dodanie parametru wyszukiwania do URL

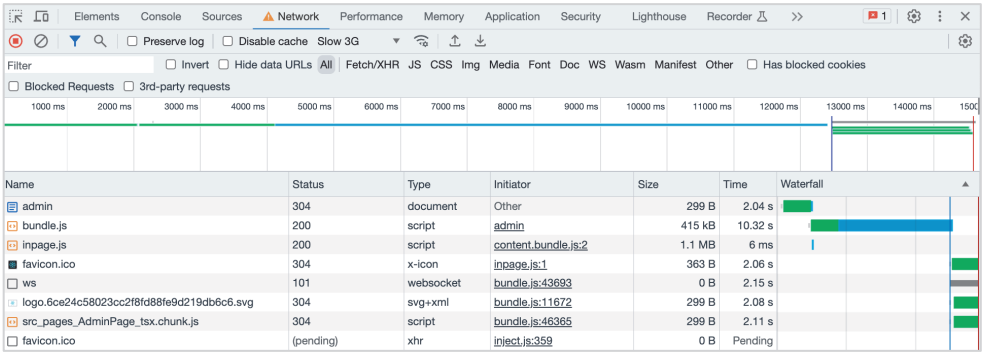
Rysunek 6.15. Lista produktów z opcją filtrowania



Rysunek 6.16. Ustawienie wolniejszego połączenia

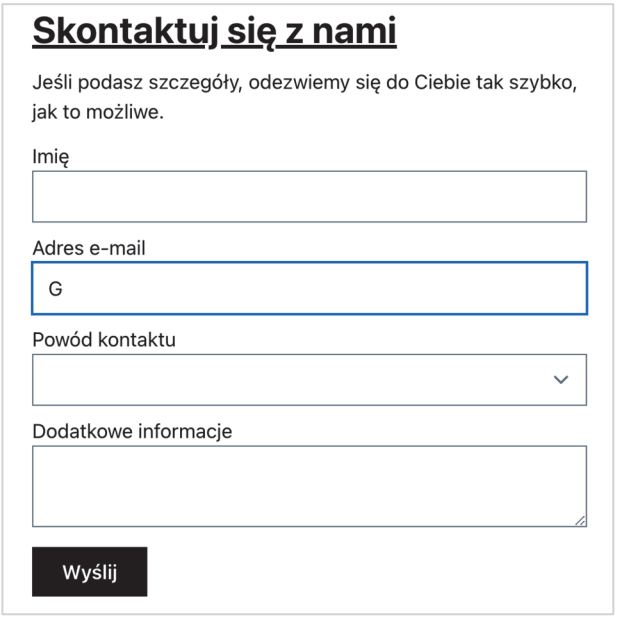


Rysunek 6.17. Wskaźnik ładowania

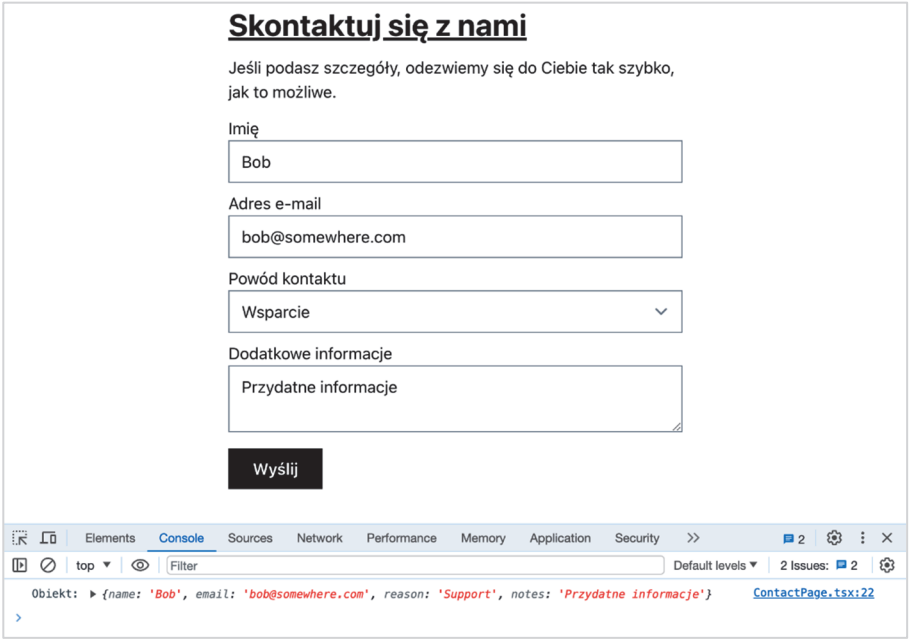


Rysunek 6.18. Proces ładowania strony administratora

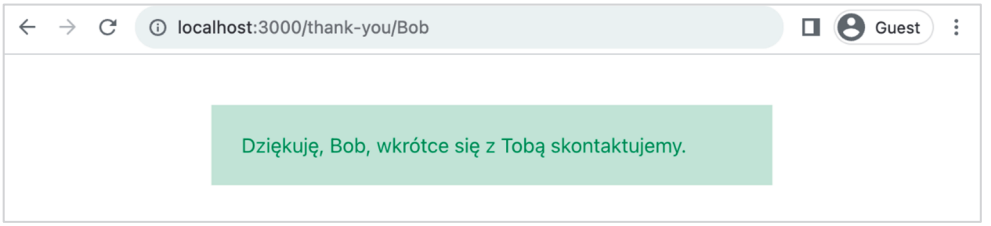
Rozdział 7. Praca z formularzami



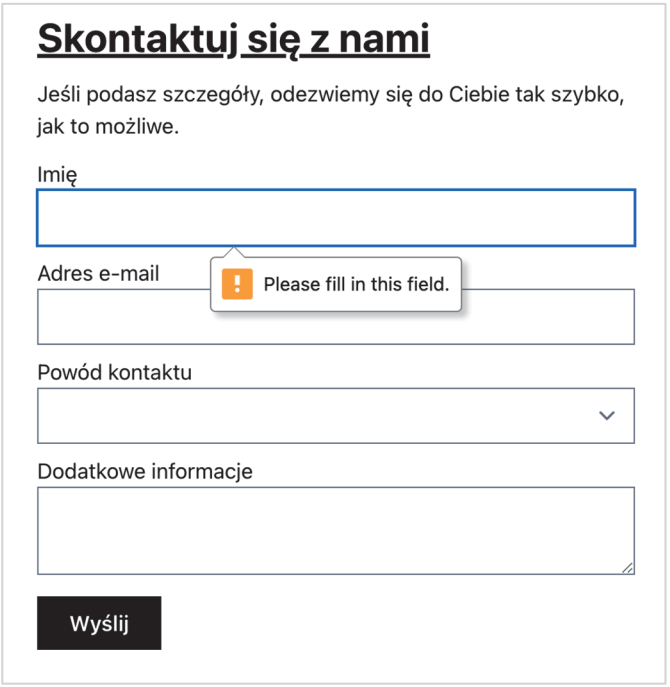
Rysunek 7.2. Podświetlenie konturu i ponowne renderowanie po każdym naciśnięciu klawisza



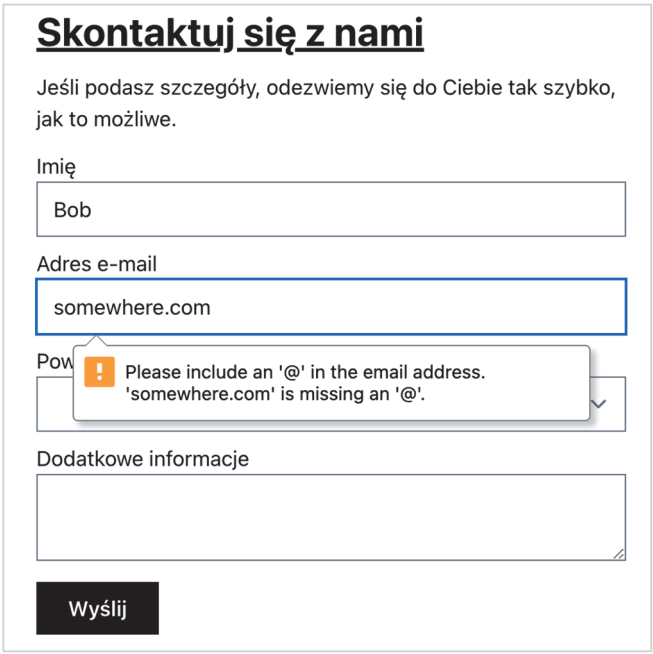
Rysunek 7.3. Kompletny formularz z danymi przesłanymi do konsoli



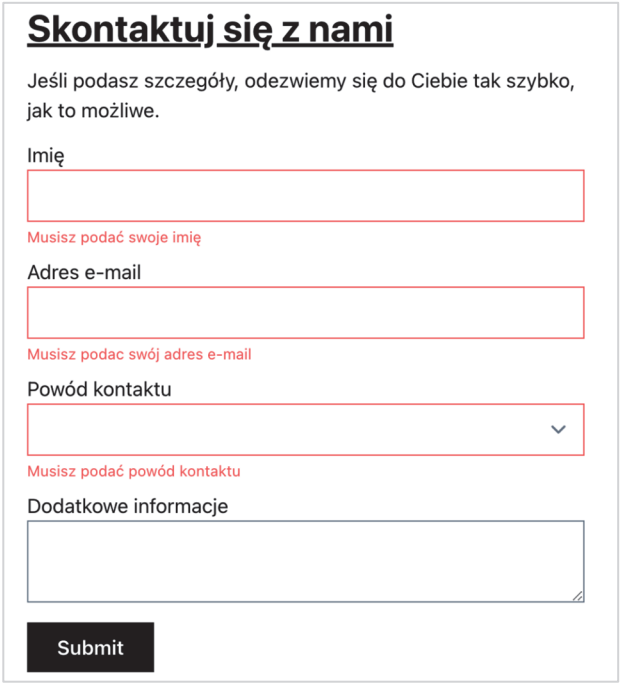
Rysunek 7.4. Strona z podziękowaniami



Rysunek 7.5. Komunikat o błędzie walidacji formularza HTML w polu na imię



Rysunek 7.6. Komunikat dotyczący walidacji formularza HTML w polu na e-mail



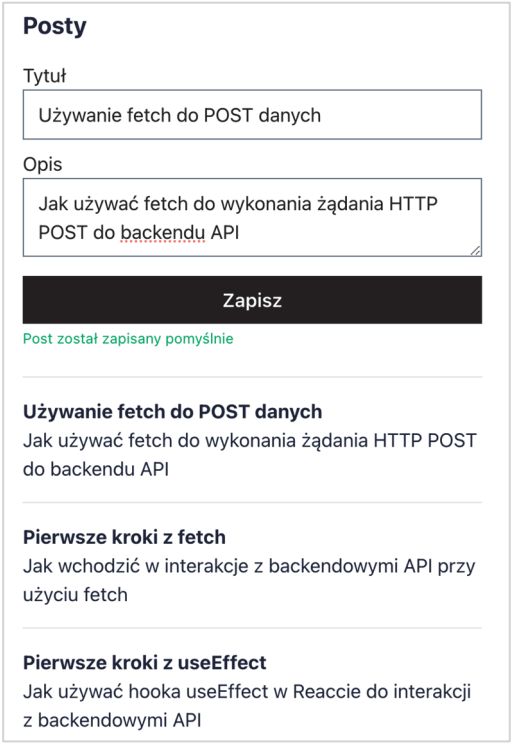
Rysunek 7.7. Zaznaczone ponowne renderowanie i błędy walidacji przy próbie wysłania formularza

Rozdział 8. Zarządzanie stanem w aplikacji

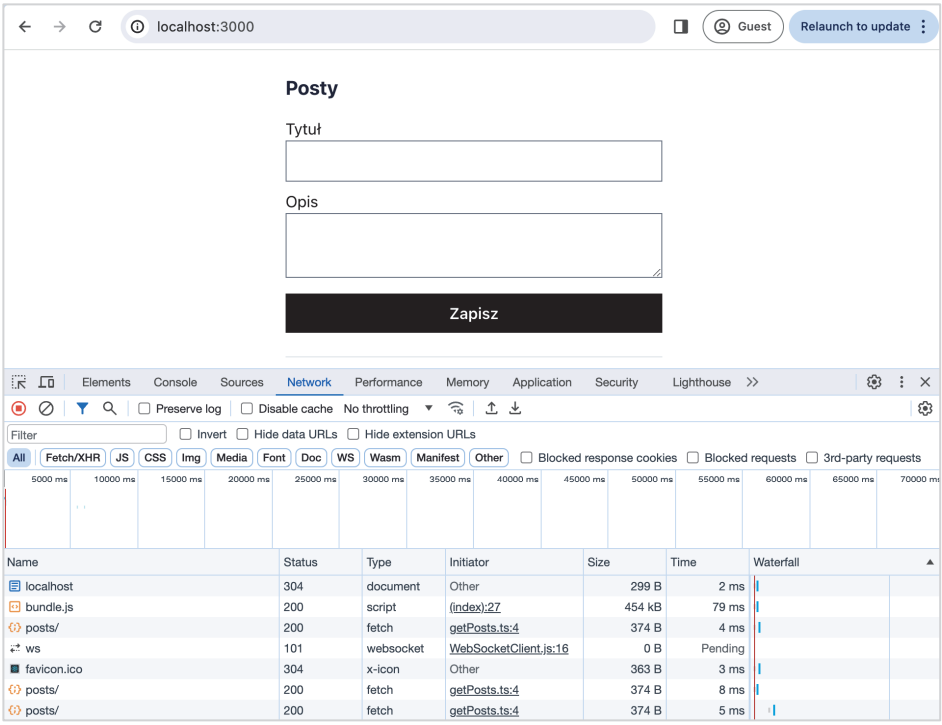
Rysunek 8.2. Wygląd aplikacji przed zalogowaniem

Rozdział 9. Praca z interfejsami RESTful API

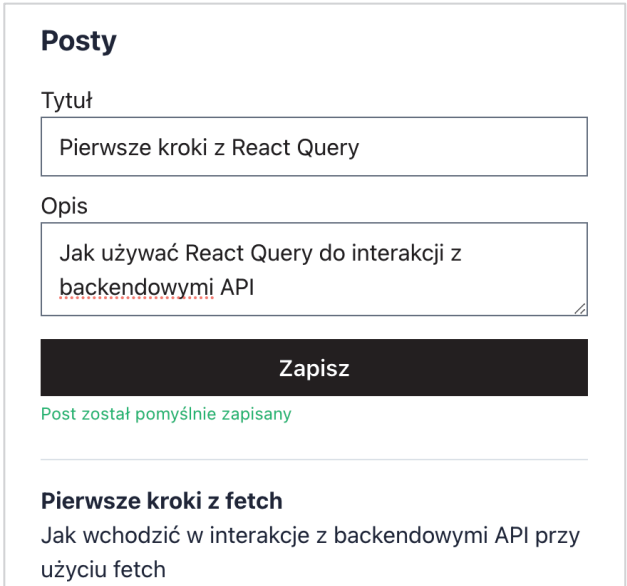
Rysunek 9.3. Wskaźnik ładowania



Rysunek 9.6. Nowy post dodany do listy

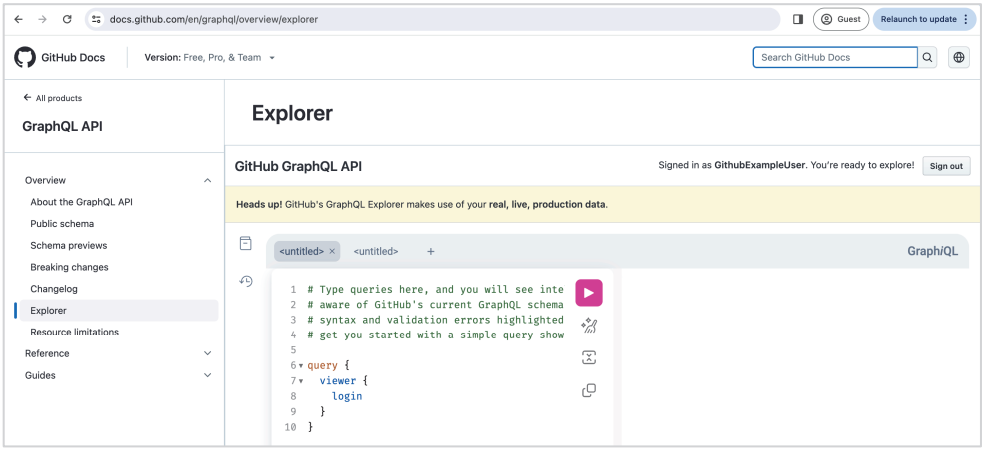


Rysunek 9.9. Dwa zapytania sieciowe dotyczące postów

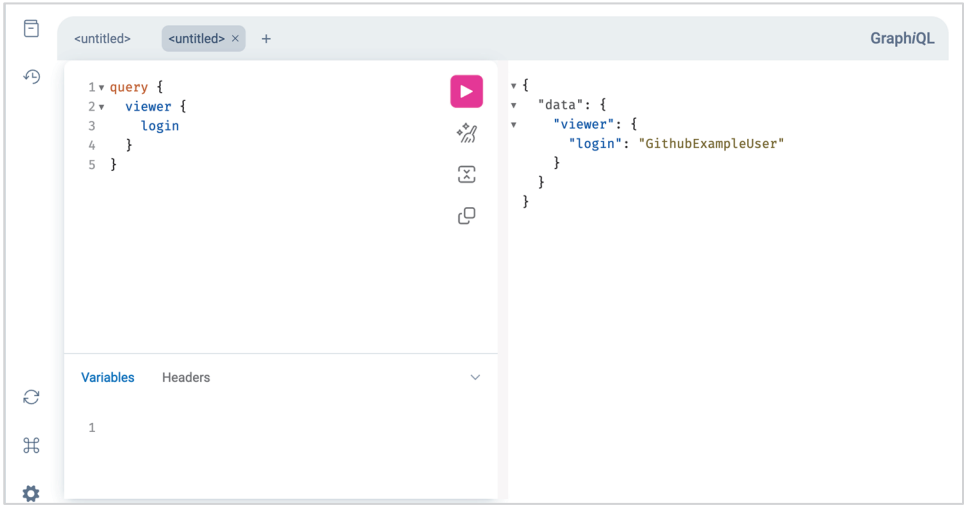


Rysunek 9.10. Nowo dodany post na liście postów

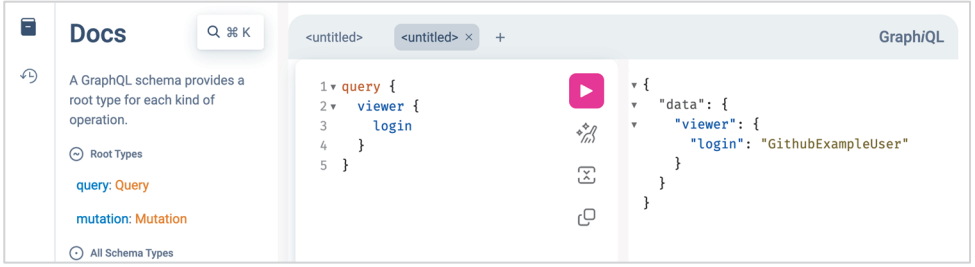
Rozdział 10. Praca z API GraphQL



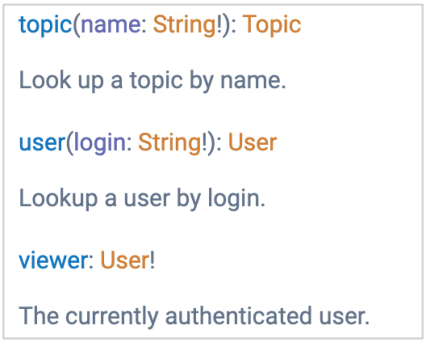
Rysunek 10.1. Eksplorator API GraphQL GitHuba



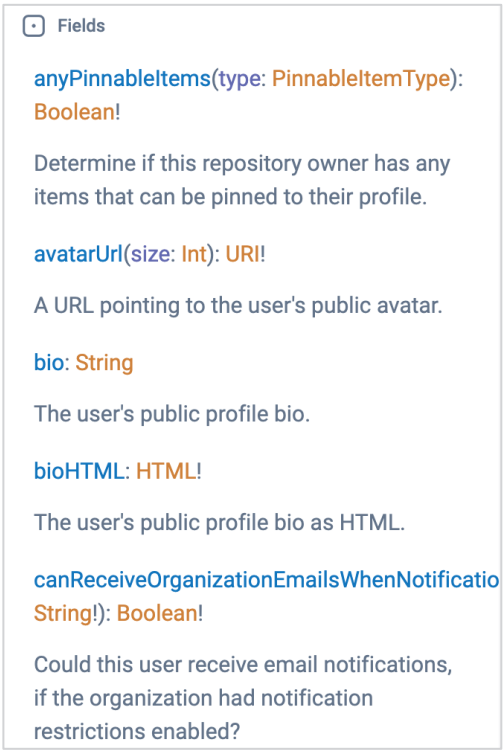
Rysunek 10.2. GitHub GraphQL



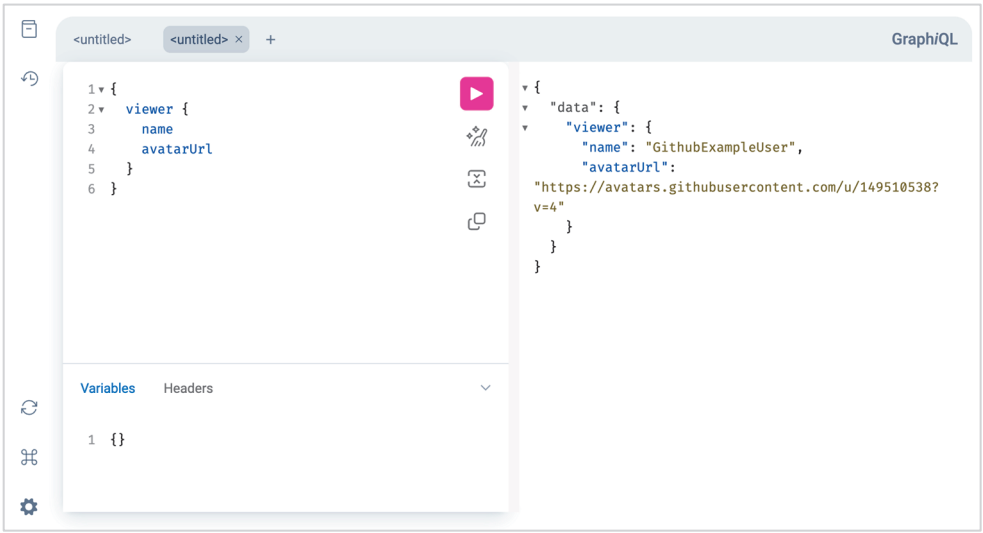
Rysunek 10.3. Panel Documentation Explorer



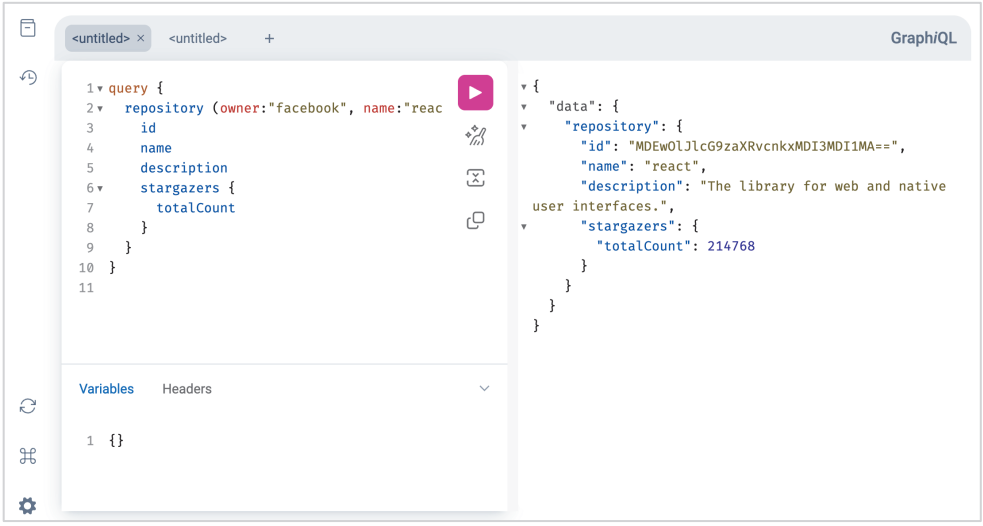
Rysunek 10.4. Obiekt viewer w eksploratorze dokumentacji



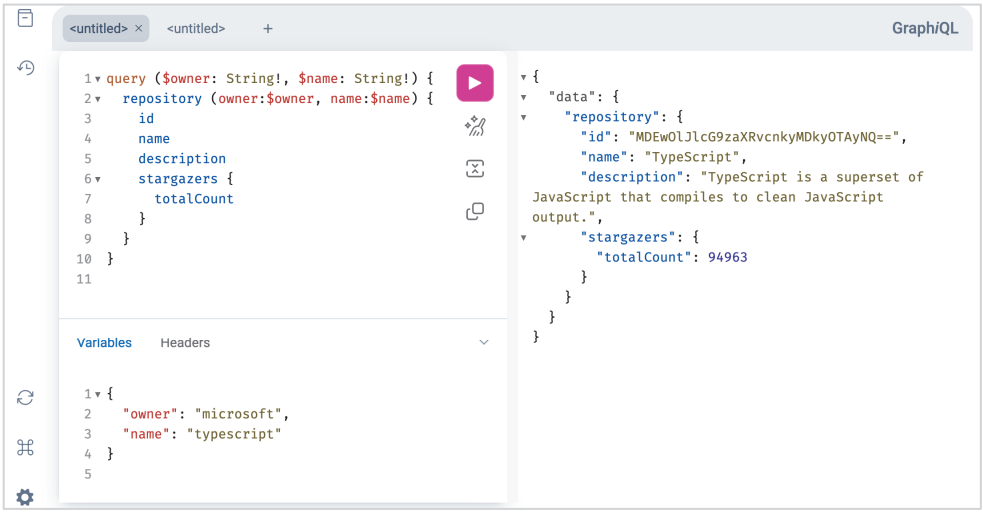
Rysunek 10.5. Lista pól dla typu User



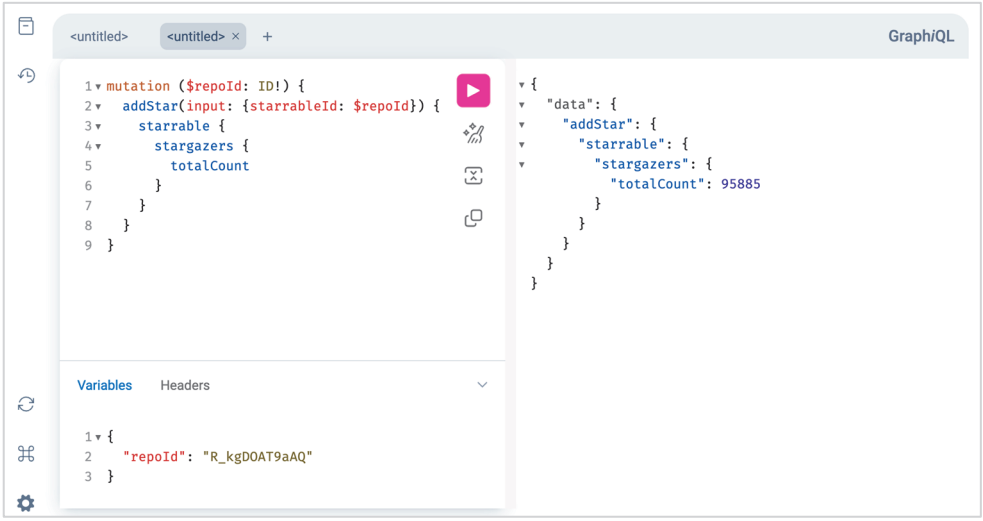
Rysunek 10.6. Zaktualizowany wynik zapytania z dodanym polem avatarUrl



Rysunek 10.7. Zapytanie o konkretne repozytorium



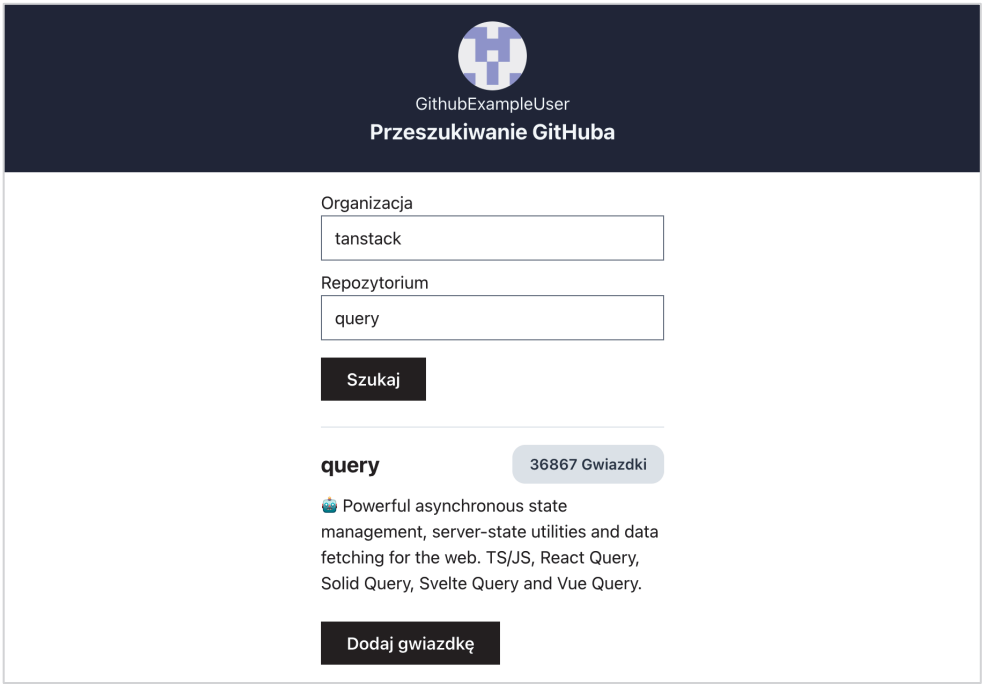
Rysunek 10.8. Zapytanie dotyczące repozytorium TypeScriptu z określonymi parametrami



Rysunek 10.9. Mutacja do oznaczania repozytorium gwiazdką

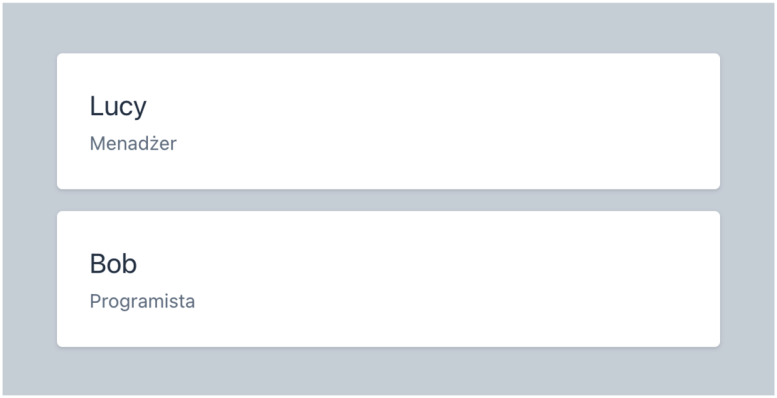
Rysunek 10.10. Nagłówek z awatarem i imieniem

Rysunek 10.12. Formularz do wyszukiwania repozytoriów

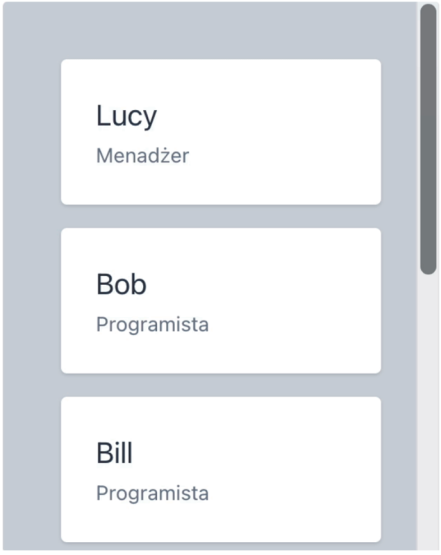


Rysunek 10.13. Znalezione repozytorium z przyciskiem Dodaj gwiazdkę

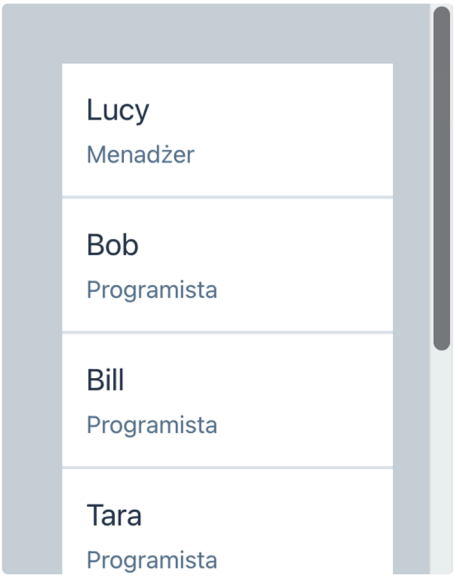
Rozdział 11. Komponenty wielokrotnego użytku



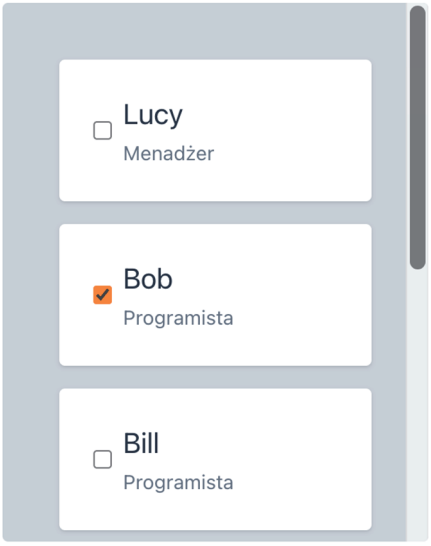
Rysunek 11.1. Prosta wersja komponentu listy



Rysunek 11.2. Dostosowana lista w komponencie



Rysunek 11.3. Zmodyfikowane elementy listy



Rysunek 11.4. Pola wyboru przy każdym elemencie

Rozdział 12. Testy jednostkowe z użyciem frameworka Jest i biblioteki React Testing Library

```
 PASS  src/Checklist/assertValueCanBeRendered.test.ts
 PASS  src/Checklist/isChecked.test.ts

Test Suites: 2 passed, 2 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        0.866 s, estimated 1 s
Ran all test suites.

Watch Usage
 › Press f to run only failed tests.
 › Press o to only run tests related to changed files.
 › Press q to quit watch mode.
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.
```

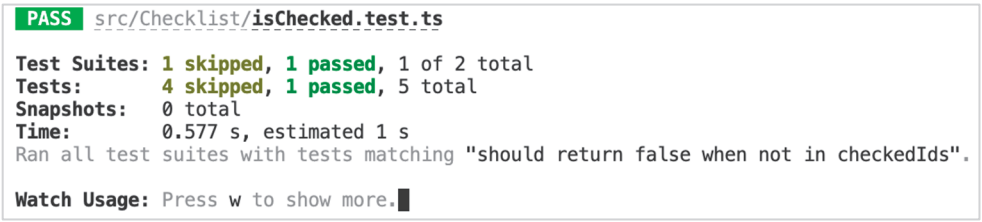
Rysunek 12.1. Pierwsze uruchomienie testów

```
 PASS  src/Checklist/isChecked.test.ts
 FAIL  src/Checklist/assertValueCanBeRendered.test.ts
  ● should raise exception when not a string or number

    expect(received).toThrow(expected)

    Expected substring: "value is not a string or a numberX"
    Received message:   "value is not a string or a number"

           5 | ): asserts value is IdValue {
           6 |   if (typeof value !== "string" && typeof value !== "number") {
         > 7 |     throw new Error("value is not a string or a number");
             |           ^
           8 |   }
           9 | }
          10 |
```

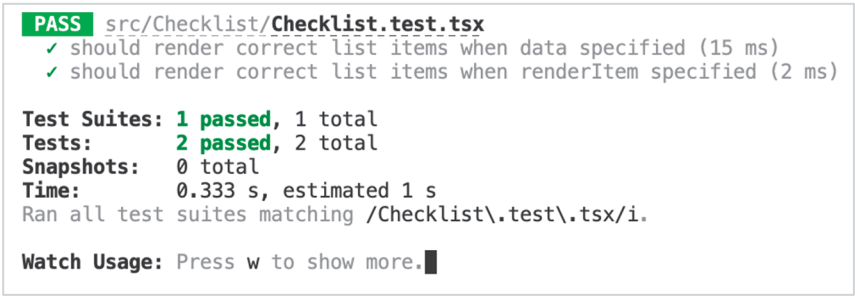
Rysunek 12.2. Nieudany test

```
 PASS  src/Checklist/isChecked.test.ts
   ✓ should return true when in checkedIds (1 ms)
   ✓ should return false when not in checkedIds

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.274 s, estimated 1 s
Ran all test suites matching /isChecked/i.

Watch Usage: Press w to show more.
```

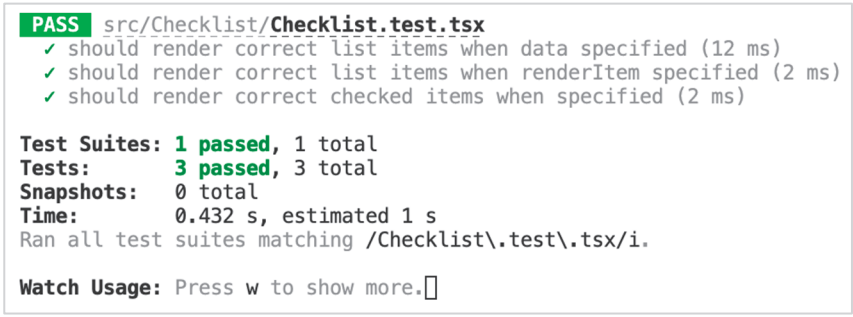
Rysunek 12.4. Jest uruchamia plik testowy w poszukiwaniu pasującego wzorca



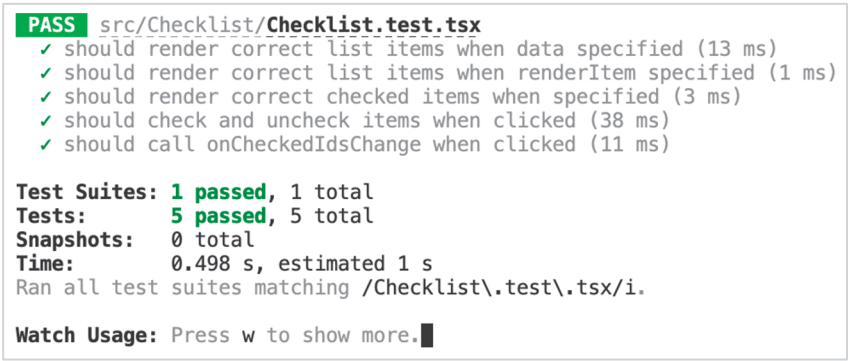
Rysunek 12.5. Jest uruchamia test o nazwie pasującej do wzorca



Rysunek 12.6. Testy komponentów zaliczone



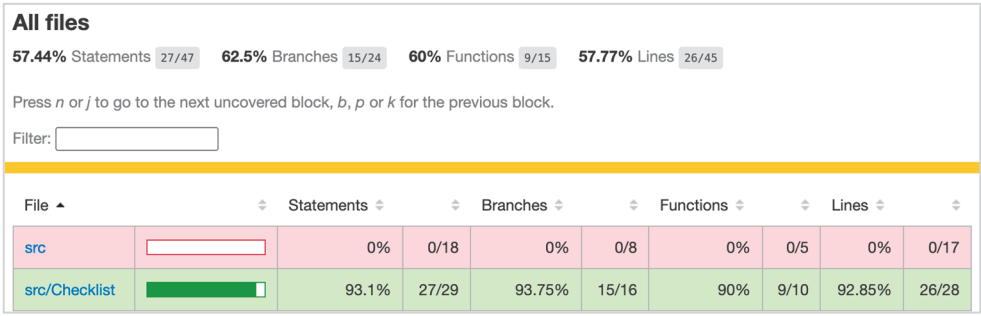
Rysunek 12.7. Wszystkie trzy testy komponentów zaliczone



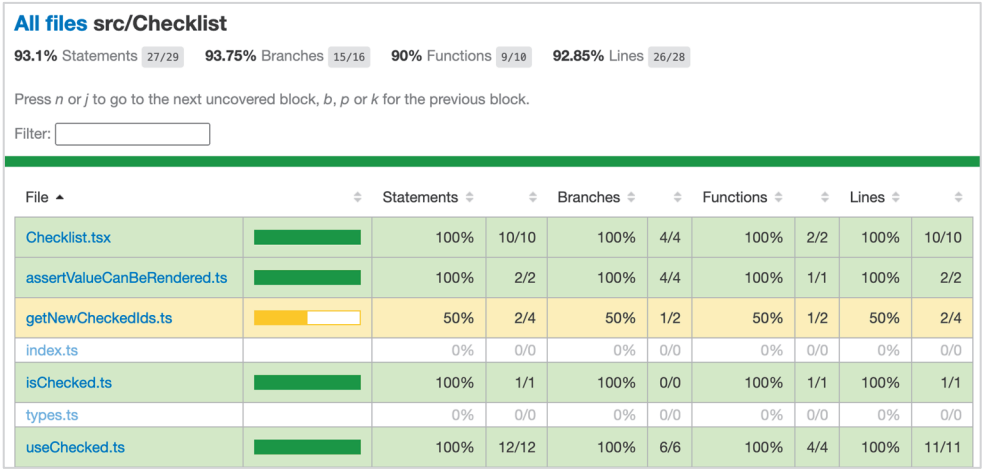
Rysunek 12.8. Pięć zaliczonych testów komponentów

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	57.44	62.5	60	57.77	
src	0	0	0	0	
App.tsx	0	0	0	0	5-14
index.tsx	0	100	100	0	7-19
reportWebVitals.ts	0	0	0	0	3-10
src/Checklist	93.1	93.75	90	92.85	
Checklist.tsx	100	100	100	100	
...ValueCanBeRendered.ts	100	100	100	100	
getNewCheckedIds.ts	50	50	50	50	9-10
index.ts	0	0	0	0	
isChecked.ts	100	100	100	100	
types.ts	0	0	0	0	
useChecked.ts	100	100	100	100	

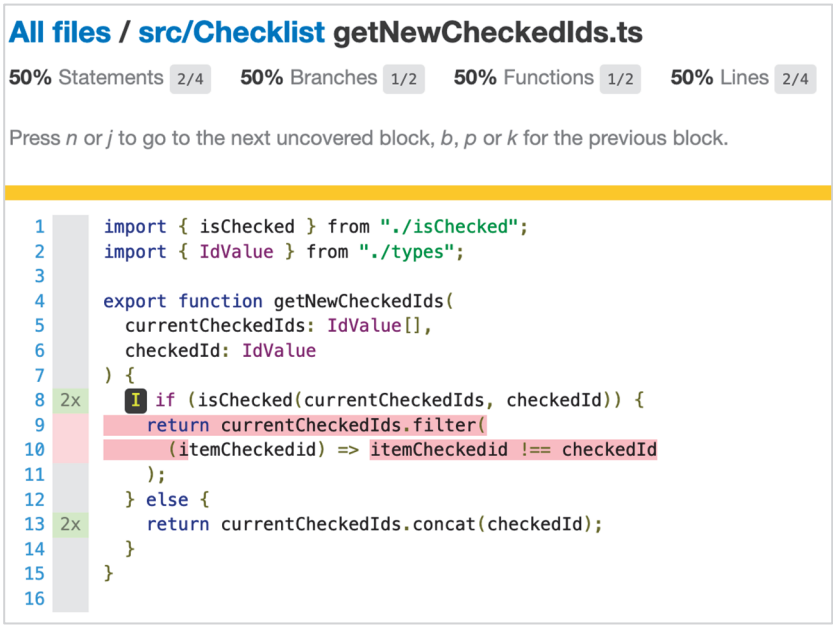
Rysunek 12.9. Raport pokrycia kodu w terminalu



Rysunek 12.10. Raport pokrycia kodu testami w HTML



Rysunek 12.11. Raport pokrycia kodu testami dla plików komponentu listy kontrolnej



Rysunek 12.12. Raport pokrycia dla pliku getNewCheckedIds.ts

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	61.7	66.66	66.66	62.22	
src	0	0	0	0	
App.tsx	0	0	0	0	5-14
index.tsx	0	100	100	0	7-19
reportWebVitals.ts	0	0	0	0	3-10
src/Checklist	100	100	100	100	
Checklist.tsx	100	100	100	100	
...ValueCanBeRendered.ts	100	100	100	100	
getNewCheckedIds.ts	100	100	100	100	
index.ts	0	0	0	0	
isChecked.ts	100	100	100	100	
types.ts	0	0	0	0	
useChecked.ts	100	100	100	100	

Rysunek 12.13. 100% pokrycia komponentu listy kontrolnej

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	65.9	66.66	66.66	66.66	
src	0	0	0	0	
App.tsx	0	0	0	0	5-14
reportWebVitals.ts	0	0	0	0	3-10
src/Checklist	100	100	100	100	
Checklist.tsx	100	100	100	100	
...ValueCanBeRendered.ts	100	100	100	100	
getNewCheckedIds.ts	100	100	100	100	
isChecked.ts	100	100	100	100	
useChecked.ts	100	100	100	100	

Rysunek 12.14. Pliki types.ts i index.ts usunięte z raportu pokrycia