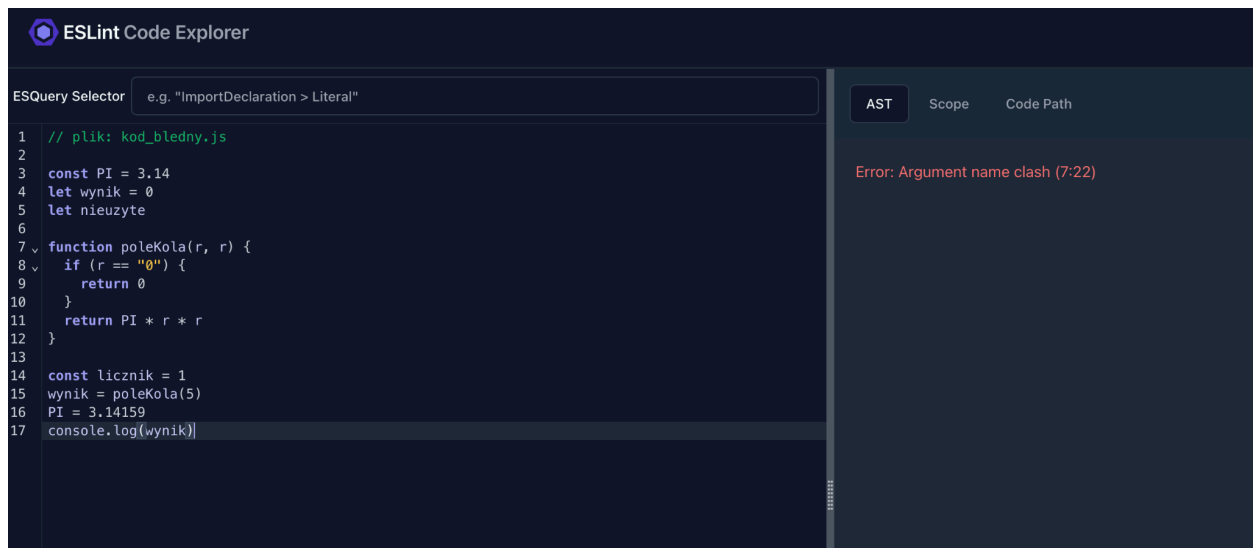


Celem ćwiczenia było sprawdzenie działania narzędzia ESLint Online (Playground) poprzez analizę prostego kodu JavaScript z błędami. Kod został napisany w programie Notepad++, a następnie wklejony do ESLint w celu wykrycia błędów i ostrzeżeń dotyczących stylu i poprawności składni.

Na początku napisałam prosty kod w pliku `kod_bledny.js`. Kod zawierał kilka błędów, między innymi zduplikowany parametr w funkcji, nieużywane zmienne, próbę zmiany wartości stałej oraz luźne porównanie `==` zamiast ścisłego `===`.

Oto użyty kod:

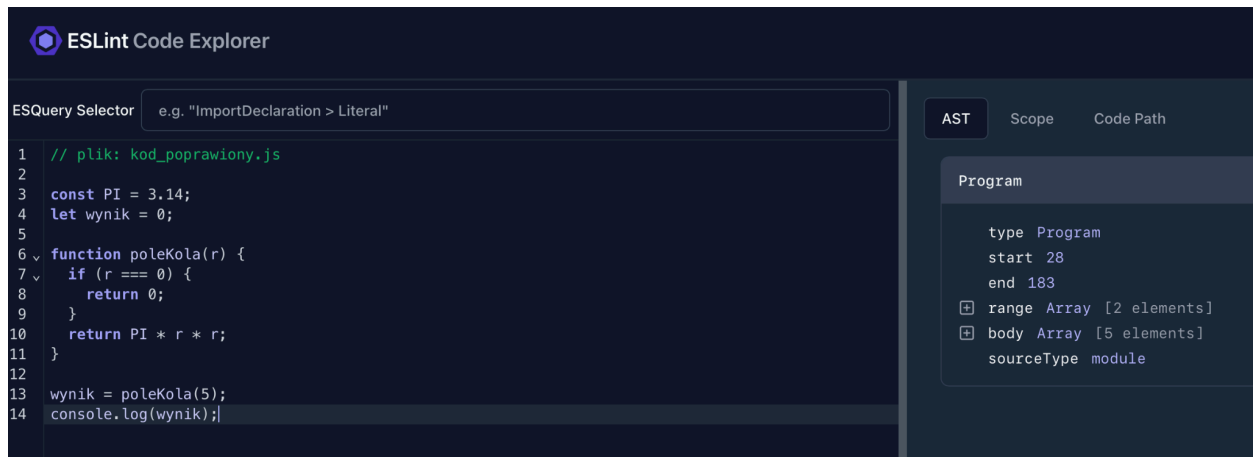


```
1 // plik: kod_bledny.js
2
3 const PI = 3.14
4 let wynik = 0
5 let nieuzyte
6
7 function poleKola(r, r) {
8   if (r == "0") {
9     return 0
10  }
11  return PI * r * r
12 }
13
14 const licznik = 1
15 wynik = poleKola(5)
16 PI = 3.14159
17 console.log(wynik)
```

Po wklejeniu kodu do **ESLint Playground** pojawiły się komunikaty błędów. Zostały wykryte następujące problemy:

- zduplikowany argument funkcji (**no-dupe-args**),
- nieużywane zmienne **nieuzyte** i **licznik** (**no-unused-vars**),
- próba zmiany stałej **PI** (**no-const-assign**),
- użycie `==` zamiast `===` (**eqeqeq**),
- brak średników (**semi**).

Następnie poprawiłam kod zgodnie z regułami ESLint. Usunęłam duplikat argumentu funkcji, nieużywane zmienne, poprawiłam operator porównania na `===`, usunęłam próbę nadpisania stałej i dodałam średniki. Po tych zmianach kod wyglądał tak:



The screenshot shows the ESLint Code Explorer interface. The main editor displays the following JavaScript code:

```
1 // plik: kod_poprawiony.js
2
3 const PI = 3.14;
4 let wynik = 0;
5
6 function poleKola(r) {
7   if (r === 0) {
8     return 0;
9   }
10  return PI * r * r;
11 }
12
13 wynik = poleKola(5);
14 console.log(wynik);
```

On the right side, the AST (Abstract Syntax Tree) view is expanded for the 'Program' node, showing the following structure:

```
type Program
start 28
end 183
range Array [2 elements]
body Array [5 elements]
sourceType module
```

Po ponownym sprawdzeniu w ESLint błędy nie były już wykrywane, a kod został uznany za poprawny.

Wynik działania programu w konsoli to **78.5**, co oznacza, że funkcja poprawnie oblicza pole koła.

Podsumowując, dzięki ESLint łatwo można wykryć błędy logiczne i stylistyczne w kodzie JavaScript. Reguły takie jak `no-dupe-args`, `no-const-assign`, `no-unused-vars` czy `eqeqeq` pomagają pisać czysty i poprawny kod. Po wprowadzeniu zmian program działał prawidłowo i spełniał standardy ESLint.