



# Patryk Jaseniuk

📍 Nysa, Polska



✉️ patryk.jaseniuk@gmail.com

🌐 LinkedIn

🐙 GitHub

🌐 <https://patrykjaseniuk.github.io/CV>

## O mnie 🧑

Witam, nazywam się Patryk, lubię władać komputerami za pomocą  **TYPESCRIPT**,  **REACT**. Aktualnie koncentruję się na pisaniu aplikacji webowych w szczególności interfejsów użytkownika. Chciałbym dołączyć do zespołu programistów i napisać coś ładnego i użytecznego.

## Projekty 📁

### Symulator Statków Żaglowych



Symulator/Gra 2D napisana w TypeScript. Użytkownik za pomocą myszki zmienia parametry żagli i steru aby kontrolować statek. Symulator prezentuje esencję działania żaglowca. Można ostrzyć, odpadać, robić zwrot przez rufę, sztag, płynąć z wiatrem, a pod wiatr tylko halsować. Cały statek jest zbudowany z molekuł, które są połączone za pomocą "sprężyn"(soft body dynamics).

### Strona Siłowni Parys



Strona reklamowa siłowni Parys w Nysie.

## Umiejętności 💪

### Języki programowania 📄



- **TypeScript** -Aktualnie mój ulubiony język programowania. Genię w aplikacjach pisanych w TS/JS łatwość uruchamiania na wielu urządzeniach (wystarczy przeglądarka internetowa, lub node). Unikam JS ze względu na brak typowania. Staram się pisać kod w stylu funkcyjnym(korzystać tylko ze stałych), ponieważ jest łatwiejszy w zrozumieniu(brak efektów ubocznych funkcji).
- **C/C++** - Pierwszy język, który poznałem. Lubilem za pomocą niego programować do czasu kiedy np. zobaczyłem jak wygodnie można posługiwać się funkcjami w TS/JS (obywatele pierwszej klasy). C++ dają więcej kontroli (brak garbage collector).

- **Java** - Drugi język który poznałem, rozwinąłem za pomocą niego koncepcję programowania obiektowego.
- **C#** - Podobny do Javy.
- **Python** - Korzystałem z niego nie zbyt dużo podczas programowania Raspberry Pi. podobny do JS (brak typowania 😞, funkcje obywatelami pierwszej klasy 😊), nie przypadło mi do gustu zastąpienie nawiasów klamrowych tabulacją.
- **PHP** - nie przepadam.

## Frameworki/ biblioteki 📚



- **React** - Po poznaniu React, polubiłem tworzenie interfejsów użytkownika.
- **NextJS** - Wygodny framework/narzędzie do budowania aplikacji TS+React.
- **Jest** - Staram się pokryć jak najwięcej kodu testami.

## Narzędzia 🛠️



- **Git** - Używam do kontroli wersji projektów, które piszę samodzielnie. Dla każdej nowej funkcjonalności tworzę nową gałąź i po zakończeniu pracy nad nią łączę ją z gałęzią główną. Staram się nazywać commit-y zgodnie z `conventional commits`.
- **Visual Studio Code** - Jest to moje ulubione IDE. Cenię jego prostotę i uniwersalność (ogromną ilość dodatków).
- **Linux** - System operacyjny o otwartym kodzie źródłowym daje znacznie większą kontrolę nad sprzętem.

## Języki naturalne 🌐

- **Angielski** - B2.
- **Polski** - Ojczysty.

## Styl programowania 📝

### Nazewnictwo

Piszę długie nazwy czasami składające się z kilku słów, unikam skrótów. Przykład: `collidingTriangle`, `FluidInteraction`

### Czytelność kodu

Podczas pisania kodu używam dużo stałych pośrednich zamiast wywoływania funkcji w funkcji. Jeżeli jestem w stanie nazwać jakiś fragment kodu(co on robi) wyodrębniam go do funkcji. Staram się nie komentować kodu, nazwy definiowane w kodzie(stałych, funkcji, interfejsów itd.) mają być wystarczające do zrozumienia.

```
//❌  
const result = doSomething(doSomethingElse(doSomethingElseAgain(doSomethingAgain())));  
  
//✅  
const somethingAgain = doSomethingAgain();  
const somethingElseAgain = doSomethingElseAgain(somethingAgain);  
const somethingElse = doSomethingElse(somethingElseAgain);  
const result = doSomething(somethingElse);
```

### Programowanie funkcyjne

Staram się pisać funkcyjnie tzn nie korzystać ze zmiennych danych ( `let` `var` ), tylko **stałych** ( `const` ). Takie założenie powoduje, że korzystanie z elementów języka z blokiem o odrębnej przestrzeni nazw (np. `for`, `if`, `switch case` ) jest bezcelowe, ponieważ, każda nazwa tam zdefiniowana jest niedostępna poza tym blokiem. Ponadto funkcje zawsze zwracają wartość, inaczej nie mają sensu.