PROGRAM SZKOLENIA DOBRE PRAKTYKI

Szkolenie Wzorce projektowe

Rozpoznasz powtarzalne problemy w kodzie i rozwiążesz je sprawdzonymi wzorcami, bez przekombinowania.

CZAS TRWANIA	POZIOM	FORMA
3 dni	**średniozaawansowany**	**warsztat dla zespołu, zdalnie lub stacjonarnie**

Ten sam problem, rozwiązany w każdym module inaczej, to częsty powód, dla którego kod trudno czytać i zmieniać. Wzorce projektowe dają Ci gotowe, sprawdzone rozwiązania i wspólne nazwy, dzięki którym zespół szybciej dogaduje się przy projektowaniu i przy code review.

Większość wzorców z książki Bandy Czworga („Design Patterns", 1994) znasz już z frameworków, tylko pod innymi nazwami: `addEventListener` to Observer, middleware to Chain of Responsibility, a wbudowany obiekt `Proxy` realizuje wzorzec Proxy bez pisania własnej klasy pośredniczącej. Na szkoleniu nazwiesz te rozwiązania, zaimplementujesz je w TypeScript i nauczysz się rozpoznawać, kiedy wzorzec upraszcza kod, a kiedy tylko dokłada warstw. Wzorce omawiasz w grupach według problemu, który rozwiązują. Najczęściej używane ćwiczysz na kodzie, a rzadsze poznajesz w module przeglądowym.

PROGRAM

3 dni

- Po co wzorce?
  - Wspólny język zespołu i katalog Bandy Czworga
  - Kompozycja zamiast dziedziczenia
  - Kiedy wzorzec pomaga, a kiedy szkodzi
- Wstrzykiwanie zależności
  - Zależności przez konstruktor i parametry funkcji
  - Kontener DI: jak działa i kiedy się opłaca
  - SOLID w skrócie: zasady, na których opiera się większość wzorców
- Tworzenie obiektów
  - Factory Method i Abstract Factory
  - Builder i Fluent API
  - Prototype: `Object.create` i własna metoda `clone()`, bo `structuredClone` gubi prototyp, a na funkcjach rzuca błąd
  - Singleton: kiedy wystarczy moduł ES i dlaczego utrudnia testy

- Dopasowanie i opakowywanie obiektów
  - Adapter (Wrapper): łączenie niezgodnych interfejsów
  - Facade: proste API przed złożonym podsystemem
  - Proxy: wbudowane `Proxy` i `Reflect`, leniwa inicjalizacja
  - Decorator: opakowanie obiektu a dekoratory TC39 (w TypeScript od 5.0, propozycja cofnięta w 2026 do Stage 2.7) i starsze `experimentalDecorators` w Angularze
- Drzewa i kolekcje
  - Composite: drzewa, np. menu i komponenty interfejsu
  - Iterator: protokół iteracji, generatory i Iterator helpers (ES2025)
- Komunikacja między obiektami
  - Observer i publish/subscribe: luźne powiązanie przez zdarzenia, a strumienie w RxJS jako rozwinięcie tego wzorca
  - `EventTarget` i `AbortController` do odpinania nasłuchu
  - Mediator: komunikacja wielu obiektów przez pośrednika (czat)
  - Chain of Responsibility: middleware w Express i interceptory
- Zmienne zachowanie
  - Strategy: wymienny algorytm, często jako zwykła funkcja
  - State: maszyna stanów zamiast rozbudowanych warunków
  - Template Method: stały szkielet algorytmu, zmienne kroki
- Cofanie operacji
  - Command: żądanie jako obiekt, kolejka operacji
  - Memento: zapis i przywracanie stanu
  - Ćwiczenie: undo/redo w edytorze
- Przegląd rzadziej używanych wzorców (bez ćwiczeń)
  - Bridge: oddzielenie abstrakcji od implementacji
  - Flyweight: współdzielenie danych przy tysiącach obiektów
  - Visitor: operacje na drzewie składni, a unia typów ze `switch` jako alternatywa
  - Interpreter: prosty język zapytań lub reguł
- Wzorce w nowoczesnym JavaScript i frameworkach
  - Które wzorce zastąpił sam język
  - Przegląd: wzorce w React (hooki, kontekst) i w Angularze (serwisy, DI)
- Antywzorce i przekombinowanie
  - Wzorzec dla samego wzorca, warstwy bez powodu
  - Ćwiczenie: refaktoryzacja nadmiarowego wzorca do prostego kodu

Zasady SOLID w pełnym wymiarze ćwiczysz na szkoleniu Clean Code, a programowanie reaktywne na szkoleniu RxJS.

DLA KOGO?

- Dla programistów, którzy widzą w kodzie te same problemy rozwiązywane za każdym razem inaczej.
- Dla zespołów, które chcą mówić o kodzie wspólnym językiem.
- Dla programistów z 2-3 latami doświadczenia, którzy projektują moduły, a nie tylko je rozwijają.

CO MUSISZ UMIEĆ?

- Swobodnie pisać w JavaScript lub TypeScript.
- Używać klas, interfejsów i modułów ES.

CZEGO SIĘ NAUCZYSZ?

- Rozpoznasz problem, do którego pasuje konkretny wzorzec.
- Wstrzykniesz zależności tak, żeby kod dało się łatwo testować.
- Zaimplementujesz najczęściej używane wzorce kreacyjne, strukturalne i czynnościowe w TypeScript i JavaScript.
- Rozpoznasz w cudzym kodzie rzadsze wzorce: Bridge, Flyweight, Visitor i Interpreter.
- Skorzystasz z mechanizmów JavaScriptu i platformy (`Proxy`, iteratory, `EventTarget`) oraz dekoratorów TypeScript zamiast pisać wzorzec od zera.
- Zastąpisz rozbudowane instrukcje warunkowe wzorcem Strategy lub State.
- Zaimplementujesz cofanie operacji wzorcami Command i Memento.
- Odróżnisz uzasadnione użycie wzorca od przekombinowania.

OPINIE UCZESTNIKÓW

> *„Pod okiem Piotra nauczysz się pisać poprawny, czytelny kod i wyćwiczysz w sobie dobre praktyki.”*
>
> Małgorzata, Frontend Developer (WarsawJS MasterClass i WarsawJS Workshop)

TRENER

Piotr Kowalski

W branży IT od 2008 roku. Od 2014 jest mentorem dla wielu developerów. Jako trener spędził na sali szkoleniowej kilka tysięcy godzin, a na LinkedIn ma 52 rekomendacje. Każde jego szkolenie to dawka solidnej wiedzy oraz humoru.

Więcej: piecioshka.pl/piotr-kowalski

ZORGANIZUJMY TO SZKOLENIE W TWOIM ZESPOLE

Program dopasuję do Waszego projektu i poziomu grupy. Napisz, z czym pracujecie, a odpowiem z propozycją terminu i zakresu.

KONTAKT **piecioshka.pl/kontakt**
E-MAIL **piecioshka+blog [małpka] gmail.com**
SZKOLENIE **piecioshka.pl/szkolenia/wzorce-projektowe/**