BO·TT·EGA
IT minds

Program szkolenia:

Architektura komponentów w React - tworzenie bibliotek i aplikacji

Informacje:

Nazwa:	**Architektura komponentów w React - tworzenie bibliotek i aplikacji**
Kod:	**react-libs**
Kategoria:	React
Odbiorcy:	developerzy, architekci
Czas trwania:	3 dni
Forma:	10% wykłady, 90% warsztaty

Warsztaty są wynikiem mojej kilkuletniej pracy połączonej z badaniami nad innymi bibliotekami o podobnym charakterze, mających te same wyzwania technologiczne. Nadszedł czas, aby podzielić się sprawdzonymi na produkcji rozwiązaniami architektonicznymi, które uważam za niesłychanie proste (ale nie łatwe!) w zastosowaniu w każdej bibliotece komponentowej.

Podczas warsztatów wykorzystamy w praktyce cztery różne podejścia do tworzenia komponentu bibliotecznego, poznamy ograniczenia i możliwości każdego z nich zarówno z perspektywy maintainera i konsumenta biblioteki. Na koniec omówimy kilka przydatnych wskazówek, które pozwolą rozplątać zależności między komponentami czyniąc je łatwiejsze w użyciu, prostsze w implementacji i bardziej elastycznymi.

WARSZTATY SĄ DLA CIEBIE, JEŚLI:

- wiesz czym są komponenty, hooki, propsy, kontekst, ale nie wiesz co dalej z tym zrobić
- zastanawiasz się, czym różni się komponent od zwykłej funkcji
- uważasz, że kontekst służy tylko do trzymania globalnych danych
- myślisz, że kontekst to tylko alternatywa dla reduxa
- używasz tzw. children inspection, ale nałożyłeś mnóstwo ograniczeń na komponent
- pomimo wszelkich starań twoje komponenty są pełne pętli i rozgałęzień
- chciałbyś pisać bardziej przewidywalny kod - odporny na zmiany wymagań oraz łatwo wnioskowany
- tworzysz bibliotekę UI, na przykład design system
- pracujesz nad systemem komponentów i chciałbyś je jakoś rozwikłać
- chcesz zagłębić się w architekturę compound components
- nawet nie tworzysz biblioteki, ale cenisz sobie dobrą architekturę

PO WARSZTATACH

- nauczysz się kilku heurystyk pomagających w tworzeniu komponentów low-coupling
- będziesz pisał kod mniej podatny na błędy
- zmienisz swoje myślenie o tym, czym jest komponent
- twoje projekty będą czytelniejsze dzięki lepszemu użyciu JSX
- nauczysz się rozwiązywać duże problemy bez tworzenia przerośniętych komponentów
- będziesz tworzył komponenty, które się “fajnie razem korzysta”
- dowiesz się, jak nie ograniczać end-usera w stosowaniu tzw. “idiomatic react”

Zalety szkolenia:

- Zajęcia praktyczne, praca na własnym kodzie, etapami - commit po commicie

- warsztaty przeznaczone dla osób, które są biegłe w tworzeniu aplikacji reactowych i interesują je zagadnienia z reużywalności kodu oraz tworzenia bibliotek ogólnego przeznaczenia (np. design system)
- mocno skupiamy się na API komponentów (piszemy w Typescript)
- pozbywamy się zbędnych abstrakcji i logiki niebiznesowej
- wyrabiamy wyczucie: co powinno być "wewnątrz" a co "poza" danym komponentem
- czysty React: tylko JSX, propsy, hooki i kontekst

BOTTEGA IT minds

Szczegółowy program:

1. Virtual DOM - podstawy

- 1.1. co to jest - pozbądźmy się mitów - to jest proste!
- 1.2. gdzie można go znaleźć - najbardziej niezrozumiany fragment Reacta
- 1.3. proces rekoncyliacji - co się dzieje między JSX a przeglądarką

2. Ćwiczenie: projekt i implementacja prostego komponentu listy

- 2.1. użycie intuicji do zbudowania komponentu bibliotecznego
- 2.2. symulacja serii pojawiających się nowych wymagań funkcjonalnych
- 2.3. iteracyjne rozbudowywanie komponentu podczas symulacji
- 2.4. obserwacja rozrostu i splątania komponentów

3. Idiomatyczny React z punktu widzenia użytkownika biblioteki reactowej

- 3.1. JSX vs POJO API
- 3.2. 4. heurystyka Nielsena

4. Stopnie splątania komponentów

- 4.1. high-coupling, no-coupling, low-coupling
- 4.2. mindset konfiguracji i mindset kompozycji
- 4.3. korzyści i konsekwencje przerzucania odpowiedzialności “na zewnątrz”

5. Ćwiczenie: Compound components - sposób na rozplątanie odpowiedzialności komponentów

- 5.1. komponenty niekomunikujące się
- 5.2. komunikacja pasywna
  - 5.2.1. powtórzenie doświadczenia implementacji listy
  - 5.2.2. props inspection
  - 5.2.3. ograniczenia virtual DOM jako źródła informacji
- 5.3. komunikacja aktywna

BOTTEGA IT minds

5.3.1. powtórzenie doświadczenia implementacji listy

5.3.2. React context

5.3.3. "Single source of truth" w DOM - czy to możliwe?

5.3.4. Compound events

5.4. porównanie poziomu skomplikowania kodu różnych podejść

6. Heurystyki rozplątywania komponentów

6.1. separacja logiki i renderingu

6.2. wielopoziomowa kompozycja

6.3. parent vs owner

6.4. przetwarzanie tablic

6.5. ReactNode vs string

6.6. zakres w callbackach

7. Ćwiczenie - refactoring przykładu z życia wziętego.

7.1. analiza problemów występujących w kodzie

7.2. użycie przedstawionych heurystyk do przebudowania realnego komponentu typu Breadcrumbs

7.3. porównanie poziomu skomplikowania kodu przed i po refactoringu

7.4. burza mózgów (live) - nowe wymagania i analiza konieczności modyfikacji kodu bibliotecznego

8. Zakończenie - praktyczne ćwiczenia z low coupling - komponenty zależne

8.1. Cel 1: komponenty wewnątrz toolbara nie mają zaokrąglonych narożników, jednocześnie nie odwołując się w kodzie toolbara do żadnego z komponentów-dzieci?

8.2. Cel 2: komponenty wewnątrz buttona automatycznie dostosowują swój rozmiar, a jednocześnie Button i PlusIcon nie muszą o sobie "wiedzieć"