Lekcja 5: Pliki deklaracji typów

Chociaż JSDoc może zaprowadzić cię bardzo daleko, czasami może być trochę nieporęczny, zwłaszcza gdy chcesz zdefiniować złożone, zagnieżdżone kształty obiektów lub chcesz ponownie użyć typów w różnych plikach. Kończysz z wieloma komentarzami i wieloma podtypami. Najprawdopodobniej bardziej zaśmieci to twoją bazę kodu, niż pomoże. Aby ułatwić nam definiowanie niestandardowych typów, opisywanie globalnych interfejsów funkcji lub współdzielenie typów między różnymi częściami naszej aplikacji JavaScript, możemy użyć plików deklaracji typów.

TypeScript, język programowania

Jest to również nasza pierwsza przygoda z językiem programowania, który bardzo przypomina JavaScript, ale w rzeczywistości jest TypeScript. Do tej pory korzystaliśmy z części narzędziowej TypeScript: narzędzia do sprawdzania typów, serwera języka, który przekazuje informacje zwrotne edytorom kodu. Nie napisaliśmy czegoś, czego nie można znaleźć w specyfikacji języka JavaScript. To, co teraz zobaczymy, jest nowe. Podobny do JavaScript, ale wystarczająco inne. Nie można go też uruchomić w przeglądarce ani w środowisku wykonawczym Node.js. TypeScript, język programowania. TypeScript nazywa się nadzbiorem języka JavaScript. Oznacza to, że TypeScript jest językiem programowania, który zawiera cały JavaScript, a także więcej konstrukcji językowych dla dodatkowych funkcji. Jednym z nich jest definiowanie niestandardowych typów obiektów.

Deklaracje typu niestandardowego

W Lekcji 4 rzuciliśmy okiem na niestandardowe typy obiektów. Zobaczmy, jak możemy zdefiniować nasz typ ShipStorage za pomocą tego nowego języka programowania:

```
type StorageItem = {

weight: number

}

type ShipStorage = {

max: number,

items: StorageItem[]

}
```

Jak widać, ta nowa deklaracja zawiera te same informacje, co nasze komentarze JSDoc w Lekcji 4. Ale sposób, w jaki definiujemy typy, wygląda zupełnie inaczej - trochę tak, jak definiujemy obiekt w JavaScript. Jeśli porównasz pamięć z naszego programu JavaScript, zobaczysz podobieństwa:

```
type ShipStorage = {

max: number,

items: StorageItem[]

}

const storage = {

max: 6000,

items: []
```

}

Dzięki deklaracji typu możemy bardzo wyraźnie określić kształt obiektów, które chcemy utworzyć. A jeśli porównasz typ i rzeczywisty obiekt obok siebie, zobaczysz, dlaczego nazywamy to kształtem. O ile piłka baseballowa ma kształt kuli, tak obiekt do przechowywania ma kształt typu ShipStorage. Mówimy też, że typ ma określoną strukturę.

Pliki .d.ts

Aby deklaracje typu działały jak ta, musimy umieścić je w pliku TypeScript. TypeScript obsługuje pliki deklaracji typu, które kończą się na .d.ts. Tutaj możesz dodać wszystkie własne typy, ale bez dodatkowego kodu programu. Przyjmujemy typy ShipStorage i StorageItem z góry i umieszczamy je w pliku types.d.ts, który znajduje się gdzieś obok głównego pliku JavaScript. Teraz mamy nasze typy w jednym miejscu. Nie są one jednak jeszcze dostępne dla innych naszych plików. Aby nasze typy były dostępne, musimy je wyeksportować:

export type StorageItem = {

weight: number

}

export type ShipStorage = {

max: number,

items: StorageItem[]

}

Pozbywamy się wszystkich deklaracji typów opartych na komentarzach w naszych głównych plikach JavaScript. Zamiast tego wskazujemy wyeksportowane typy. Zaraz po komentarzu @ ts-check dodaj następujące dwie linie:

/** @typedef { import('./types.d').ShipStorage }

ShipStorage */

/** @typedef { import('./types.d').StorageItem }

StorageItem */

Składnia jest bardzo podobna do naszej poprzedniej definicji typu. Ale zamiast mówić TypeScript - lub JSDoc - że definiujemy obiekt, wskazujemy bezpośrednio na obiekt, który już zdefiniowaliśmy gdzie indziej. Właśnie napisaliśmy nasze pierwsze niestandardowe typy w języku TypeScript! Należy pamiętać, że nasz kod JavaScript w ogóle się nie zmienił; nadal piszemy zwykły JavaScript dla rzeczywistego rdzenia naszego programu. Definicje typów języka TypeScript po prostu istnieją z boku, co ułatwia życie. Jest to jedna z podstawowych zasad języka TypeScript: maksymalne ułatwienie dodawania typów do kodu programistycznego. Jak dotąd nie potrzebujesz żadnych narzędzi, tylko edytor, który wie, jak obsługiwać TypeScript. Prowadzi to do stopniowego i dyskretnego przepływu pracy, który umożliwia dodawanie języka TypeScript bez zbytniego angażowania się w narzędzia i procesy kompilacji.

1. Za pomocą // @ ts-check aktywujemy TypeScript w aktualnie edytowanym pliku.

2. Używamy adnotacji typu komentarza JSDoc dla wszystkich naszych stałych, obiektów i funkcji. Zwłaszcza funkcje czerpią duże korzyści z dodatkowych informacji.

3. Tworzymy niestandardowe definicje typów w plikach deklaracji typów i ładujemy je w razie potrzeby w naszych adnotacjach typu JSDoc.

Ten sposób tworzenia języka TypeScript nie jest rzadkością. To świetny sposób na stopniową migrację do TypeScript, ale jednocześnie wystarczający, aby w pełni wykorzystać zalety TypeScript bez zrażania współtwórców poprzez wprowadzenie nowego języka programowania. Jednym z popularnych frameworków wykorzystujących to podejście jest Preact. Cała baza kodu Preact opiera się na dodanych typach!