Wybrane elementy praktyki projektowania oprogramowania
Zestaw 6

TypeScript

2025-11-18

Liczba punktów do zdobycia: **10/55**
Zestaw ważny do: 2025-12-02

1. (**1p**) Użyć kompilatora TypeScript z linii poleceń oraz w trybie **watch** do kompilowania przykładowego kodu. Pokazać że wytworzony w ten sposób kod można debugować z poziomu Visual Studio Code.

   Użyć **tsx**, **ts-node** lub innego transpilera do uruchamiania kodu TypeScript. Pokazać że w tym przypadku kod również można debugować z poziomu Visual Studio Code.

   Lista dostępnych transpilerów:

   `https://github.com/privatenumber/ts-runtime-comparison`

   Użyć natywnie node.js w wersji 24 do uruchomienia kodu TypeScript. Pokazać że tu również można debugować kod z poziomu Visual Studio Code.

2. (**1p**) Przystosować napisaną przy okazji jednego z poprzednich zestawów rekurencyjną implementację funkcji `fib(n)` i funkcję memoizującą `memoize` do TypeScript.

   Uwaga! Zadanie ma wiele rozwiązań, w zależności od tego jakie sygnatury typów przypisze się funkcjom **fib** i **memoize** kontrola typów będzie silniejsza lub słabsza. Znaleźć jakiś kompromis między użytecznością, a skutecznością.

3. (**1p**) Dane są typy

```
type Person = {
  name: string,
  surname: string
}

type Animal = {
  name: string,
  species: string
}
```

   Czym są poniższe typy? Napisać funkcje które przyjmują obiekty tych typów i pokazać jak je wywołać z prawidłowymi argumentami.

```
type PersonAndAnimal = Person & Animal;
type PersonAndString = Person & string;
type PersonOrAnimal  = Person | Animal;
type PersonOrString  = Person | string;
type StringAndNumber = string & number;
type StringOrNumber  = string | number;
```

4. (**1p**) Dane są typy

```
type User = {
    name: string;
    age: number;
    occupation: string;
}

type Admin = {
    name: string;
    age: number;
    role: string;
}

type Person = User | Admin;

const persons: Person[] = [
    {
        name: 'Jan Kowalski',
        age: 17,
        occupation: 'Student'
    },
    {
        name: 'Tomasz Malinowski',
        age: 20,
        role: 'Administrator'
    }
];
```

oraz funkcja

```
function logPerson(person: Person) {
    let additionalInformation: string;
    if (person.role) {
        additionalInformation = person.role;
    } else {
        additionalInformation = person.occupation;
    }
    console.log(` - ${person.name}, ${person.age}, ${additionalInformation}`);
}
```

która w takiej formie jak wyżej nie skompiluje się - na ścieżkach warunkowych dla **if** kompilator nie jest w stanie zawęzić typu parametru do jednego z dwóch podanych. Jak skorygować powyższy kod tak żeby poprawnie się skompilował i zadziałał?

Wskazówka: zastosować technikę unii z wariantami (rozdział 4.9 w notatkach).

5. (**1p**) Do poprzedniego zadania dodano pomocnicze funkcje pełniące rolę strażników typowych (choć do rozwiązania poprzedniego zadania to nie było konieczne!). Należy skorygować definicje pomocniczych funkcji **isAdmin** i **isPerson** tak żeby faktycznie pełniły rolę strażników typowych w funkcji **logPerson**.

```
function isAdmin(person: Person) {
    return ???;
}

function isUser(person: Person) {
    return ???;
}

function logPerson(person: Person) {
    let additionalInformation: string = '';
    if (isAdmin(person)) {
        additionalInformation = person.role;
```

```
    }
    if (isUser(person)) {
        additionalInformation = person.occupation;
    }
    console.log(` - ${person.name}, ${person.age}, ${additionalInformation}`);
}
```

6. (**1p**) Pokazać na własnych przykładach jak używać następujących typów:

   - typy wyższego rzędu - `Extract`, `Exclude`
   - typy wyższego rzędu - `Record`, `Required`, `Readonly`, `Partial`
   - typy wyższego rzędu - `Pick`, `Omit`
   - typy indeksowane (Indexed Acess Types) (rozdział 4.11 w notatkach)

   Jakie jest zastosowanie dla tych mechanizmów? Jaką wartość wnoszą do systemu typów?

7. (**4p**) Rozwiązać wszystkie zadania z sekcji **Warm up** i **Easy** ze zbioru Type Challenge. Opcjonalnie (dla własnej satysfakcji, bez dodatkowych punktów), rozwiązać wybrane samodzielnie zadania z pozostałych sekcji (zadania z sekcji Hard i Extreme są naprawdę trudne!).

   `https://github.com/type-challenges/type-challenges`

   Uwaga! Każdy z przykładów ma łatwo dostępne rozwiązania, w zadaniu nie chodzi więc o to żeby się wykazać umiejętnością wyszukania gotowego rozwiązania, ale żeby coś z tego wynieść, czyli spróbować mimo wszystko zmierzyć się z tym samodzielnie, a dopiero w ostateczności popatrzeć na prawidłowe rozwiązanie i spróbować je zrozumieć.

Wiktor Zychla