TypeScript

Vanderkam Dan, “Effective TypeScript – 62 Specific Ways to Improve Your TypeScript”, O’Reilly Media, 2020

Intre TypeScript si JavaScript exista o dependenta foarte mare, dar care poate crea confuzii. De pilda, compilarea acestui limbaj de programare – TypeScript - are loc in cadrul JavaScript.

Se spune ca TypeScript este un superset al Javascript, adica orice cod JavaScript devine cod Typescript prin redenumirea extensiei, insa exista cod scris in TypeScript care nu este cod JavaScript pentru ca TypeScript are si sintaxa proprie. Fisierele corespunzatoare pentru TypeScript au extensia .ts sau .tsx, in timp ce fisierele corespunzatoare pentru JavaScript au extensia .js sau .jsx, si totusi codul scris in fisirele tip JavaScript poate fi deja interpretat in cadrul TypeScript.

Un avantaj al TypeScript este posibilitatea de a detecta secvente de cod care ar duce la exceptii la rulare, cat si acele secvente de cod care nu ar indeplini scopul pentru care au fost scrise, astfel ca la scrierea programelor in TypeScript se incearca a se trece toate aceste verificari facute in mod automat. Totusi, se poate ca un program sa compileze chiar daca exista unele erori de scriere.

Configurarile pentru TypeScript se scriu fie in linia de comanda, fie intr-un fisier de configurare numit tsconfig.json, fiind recomandat a se folosi fisierul de configurare astfel incat si alti colegi sa poata intelege cum va fi folosit TypeScript.

De-asemenea, este important editorul ales, astfel incat sa permita folosirea serviciilor oferite de TypeScript, precum sugestiile in timp ce este scris codul. Sunt de preferat declararile de forma tip “:Type” fata de declararile de tip “as Type”. Se recomanda si folosirea primitivelor tipurilor de variabile in locul obiectelor de tip wrapper, de pilda “number” in loc de “Number”.

Vilic Vane, “TypeScript Design Patterns”, Packt Publishing, 2016

Pentru a putea folosi TypeScript este necesar a se instala Node.js, cat si compilatorul pentru TypeScript. Pentru a verifica daca Node.js a fost instalat correct, se folosesc comenzile:

* *node -v*
* *npm - v*

Pentru a instala compilatorul de TypeScript, se foloseste comanda:

* *npm install typescript -g*

Exista numeroase editoare care ajuta la realizarea programelor in TypeScript, precum Visual Studio Code sau Sublime, care sunt cele mai recomandate, dar se pot folosi si Atom, WebStorm sau IntelliJ.

Fisierul de configurare tsconfig.json accepta urmatoarele trei campuri: “compilerOptions”, “files” si “exclude”.

Se pot folosi si fisiere de declarare, care ajuta la folosirea librariilor JavaScript existente. Pentru a le descarca, se foloseste pachetul Node.js numit “typings”, care se instaleaza si apoi verifica utilizand comenzile:

* *npm install typings -g*
* *typings --version*

Fisierele de declarare sunt necesare, intrucat fara ele compilatorul ar returna o eroare, iar API-urile corespunzatoare Node.js nici nu ar putea fi folosite. Ele se salveaza in folderul “typings”.

Testarea codului este foarte importanta si se poate realiza cu Mocha si Instanbul. Mocha este cel mai cunoscut framework de testare pentru JavaScript, iar Chai se foloseste des ca librarie pentru validari. Istanbul se foloseste pentru vedea cat la suta din cod este acoperit de teste. Acestea din urma, pot fi folosite dupa rularea comenzilor:

* *npm install mocha -g*
* *npm install chai --save-dev*
* *npm install istanbul -g*

In TypeScript exista numeroase modele de design de creare al codului, precum “builder”, “singleton” sau “prototype”. De pilda, in cazul “builder” se definesc pasii necesari pentru a construi obiecte complexe, iar apoi fie se schimba ordinea pasilor fi se foloseste o alta implementare de tip “builder”. In cazul “singleton”, e necesar a se creea cate o singura instanta.

In plus, exista numeroase modele de design al structurii codului. Printre acestea se numera modelele “Decorator”, “Bridge”, “Flyweight” si altele. De exemplu, in cazul modelului “Flyweight” obiectele de tip “stateless” care sunt folosite de multe ori sunt impartite intr-o maniera ce imbunatateste performanta si eficienta memoriei.

Foarte utilizate sunt si modelele de design al comportarii codului, adica ajuta la stabilirea modului in care obiectele si clasele interactioneaza intre ele. Cateva dintre cele mai utilizate modele de design al comportarii codului sunt “Command”, “Iterator” sau “Mediator”. In cazul “Mediator”, obiectele inrudite logic sunt grupate, iar acest lucru ajuta la a avea un sistem cu care se poate interactiona mai usor.

React Native