»Mladi za napredek Maribora 2020«

37. srečanje

Interpretacija skriptnega programskega jezika JavaScript

RAČUNALNIŠTVO IN INFORMATIKA

Raziskovalna naloga

PROSTOR ZA NALEPKO

Avtor: NEJC DROBNIČ

Mentor: BRANKO POTISK
Šola: SREDNJA ELEKTRO-RAČUNALNIŠKA ŠOLA MARIBOR
Število točk: 72/ 170

Maribor, februar 2020

»Mladi za napredek Maribora 2020«

37. srečanje

Interterpretacija skriptnega programskega jezika JavaScript

RAČUNALNIŠTVO IN INFORMATIKA

Raziskovalna naloga

PROSTOR ZA NALEPKO

Maribor, februar 2020

KAZALO

KAZALO SLIK

1 POVZETEK

Naloga uspešno predstavi možnosti interpretacije in razvijanje »transpilerjev« za pretvorbo programskega skriptnega jezika TypeScript oz. JavaScript.

Velik bonus natančne interpretacije in prikaza njenih podatkov ter »transpilacije« iz teh podatkov v drug programski jezik omogoča aktivno opazovanje razlik jezikov in s tem omogoča hitrejšo in lažje učenje iz primerov ali surovih interpretacijskih podatkov v strukturi AST oz. ASD.

2 UVOD

Namen te raziskovalne naloge je predstaviti možnosti interpretacije programskega skriptnega jezika JavaScript in ene izmed največjih variacij, ki je nastaja na osnovi jezika. V nalogi smo načrtovali narediti prototip programa za razstavo kode v njene sestavne dele, da bi lahko omogočilo »transpilacijo« iz jezika JavaScript v druge programske jezike.

3 CILJ NALOGE

Naša raziskovalna naloge je imela sledeče cilje:

- Preveriti zmožnosti interpretacije programskega skriptnega jezika JavaScript.
- Preverite alternativne metode procesiranja in prevajanja kode programskega skriptnega jezika JavaScript ali katerega koli izmed njegovih super-setov.
- Raziskati potencialne uporabe principa v učnem procesu ali mentorstvu.

4 POTREBNO PREDZNANJE

4.1 LEGENDA

Kratica	V Angleščini	Opis
AST	Abstract Syntax Tree	Abstract Syntax Tree oz. Abstraktno Sintaksno drevo ali Syntax Tree oz. Sintaksno drevo je drevesna reprezentacija abstraktne sintakse izvirne kode.
JS	JavaScript	Programski jezik zasnovan za uporaba na spletu. Prvič se je pojavil 4. 12. 1995.
TS	TypeScript	Super-set programskega jezika JavaScript, ki vključuje striktno primerjavo tipov. Prvič se je pojavil 1. 10. 2012.
JSON	JavaScript Object Notatation	

4.2 »casting«

V TS je »casting« dejanje s katerim lahko jeziku rečeš, da je tip nečesa drugačen od tega, kaj razbere oz. pričakuje. Primer: »value« as »type« .

4.3 Izvirna Koda

Izvirna koda projekta bo ob začetku javnih zagovorov objavljena na GitHub-u in dostopna z sledečo povezavo https://github.com/TypedTo/TypeToAST .

5 METODOLOGIJA DELA

V procesu

5.1 JavaScript

```
class Naloga {

  constructor(ime) {
    this._ime = ime;
  }

  get ime() {
    return _ime;
  }

}

class RaziskovalnaNaloga extends Naloga {

  constructor(ime, tema) {
    super(ime);

    if (tema) this._tema = tema;
  }

  setTema(tema) {
    this._tema = tema;
  }

  get tema() {
    return this._tema;
  }

}

const raziskovalna = new RaziskovalnaNaloga('Interpretacija skriptnega programskega jezika JavaScript');

// Vse deluje saj ni nobene omejitve
raziskovalna.setTema('racunalnistvo');
raziskovalna.setTema('tehnika');
```

Slika 1 Primer splošne JavaScript kode

Zaradi omejitev jezika v zvezi z tipi ter parserji smo se odločili nalogo nadaljevati v programskem jeziku TypeScript, ki ga razvija Microsoft in ponuja odlično podpira tipov ter že vgrajen parser oz. »compiler«. S to ugotivitvijo se je naša uporaba jezika JavaScript zaključila saj ni zadoščalo našim potrebam.

5.2 TypeScript

```typescript
class Naloga {
  readonly _ime: string;

  public constructor(ime: string) {
    this._ime = ime;
  }

  public get ime(): string {
    return _ime;
  }

}

class RaziskovalnaNaloga<T extends string> extends Naloga {
  protected _tema: T;

  public constructor(ime: string, tema?: T) {
    super(ime);

    if (tema) this._tema = tema;
  }

  public setTema(tema: T) {
    this._tema = tema;
  }

  public get tema() {
    return this._tema;
  }

}

const raziskovalna: RaziskovalnaNaloga<'racunalnistvo' | 'kemija' | 'informatika'>
      = new RaziskovalnaNaloga('Interpretacija skriptnega programskega jezika JavaScript');

raziskovalna.setTema('racunalnistvo'); // Deluje saj se podana vrednost ujema z dovoljenimi

raziskovalna.setTema('tehnika'); // Ne deluje saj se podana vrednost ne ujema z dovoljenimi
```

Slika 2 Primer splošne TypeScript kode

Je nadgrajena verzija programskega skriptnega jezika JavaScript saj doda funkcijo enumeracijskih objektov, privatne, javne, samo zapisne in zavarovane funkcije v razrede.

Ter določene beta vsebine, ki so še v TC39 »proposal« postopkih. TypeScript razvija podjetje Microsofta in prva javna verzije se je pojavila 1. oktobra 2012.

TS ima tudi priložen surov dostop do svojega »transpiler«-ja. To dejstvo je omogočilo gladek potek naloge. Saj je to pomenilo, da smo delali z orodjem, ki ga je uporabljal sam jezik.

6 PODROBNEJŠI OPIS DELOVANJA

```
import * as ts from 'typescript';
import { GenerateSchema } from './lib/GenerateSchema';

const schema = new GenerateSchema(process.argv.slice(2), {
    target: ts.ScriptTarget.ES5,
    module: ts.ModuleKind.CommonJS
});
schema.compile();

schema
    .write()
    .then(() => console.log('Completed Schema Generation'))
    .catch(() => null);
```

Slika 3 Primer uporabljenega prototipa

V zgornji sliki lahko vidimo uporabo prototipa programa, kjer mu podamo drug argument komande »node«, npr. »node ./dist/index.js ./test/koda.ts«.

Program deluje tako da prebere datoteko in jo poda TS prevajalniku, ki začne programatično brati kodo in izdelovati AST oz. Abstraktno Sintaksno Drevo. Seveda če pa uporabnik poda definicijsko datoteko (datoteka s pripono ».d.ts«) jo preskočimo, saj za ta del postopka nima pomena. Temu za vsak element zaženemo funkcijo, ki nato »obišče« vsak element in zažene drugo funkcijo, ki je prirejena za dano strukturo. Iz te nato naredimo simbol, da lahko dostopamo do več informacij nakar celoten simbol potisnemo v list za izpis, ki je pozneje uporabljen za zapis shematske JSON strukture.

7 REZULTATI

```
/**
 * Documentation for C
 */
class C {
    /**
     * constructor documentation
     * @param a my parameter documentation
     * @param b another parameter documentation
     */
    public constructor(a: Error, b: C, c: int, d: aaaa) {
    }

    public tuturu(): string {
        return 'tuturu';
    }

}

function add(a: int, b: ushort): uint {
    return a + b;
}
```

Slika 4 Testna koda

V prvi slika na desni strani je vidno kaj smo dali v shema generator. Kot je vidno je podana koda v jeziku TS. V naslednji sliki se lahko vidi kako naš program oz. prototip spremeni kodo strukturno shemo, ki se lahko nato npr. uporabi v prevajalniku oz. »transpilerju«.

Slika 5 AST Izvoz testne kode

Prvi objekt v listu predstavlja strukturo razreda vključno z vsemi podatki o definiciji kot so ime, dokumentacija, tip, metode ter definicijo konstruktorja, ki vključuje definicije parametrov, ki so opremljeni spet z dokumentacijo, imenom in tipom; medtem ko drugi predstavlja dodano funkcijo »add«. Vsebuje enake podobne podatke kot definicija konstruktorje le z drugačno ureditvijo.

V celote list predstavlja vse strukturne definicije in njihove podatke, ki so bili prisotni v sliki št. 4.

8 INTERPRETACIJA REZULTATOV

Iz podanega rezultata testnega programa smo lahko razbrali, da je za pridobivanje določenih podatkov potrebno zelo dosti infrastrukture, ki izhaja iz ekstreme vrnjenih podatkov in v tem kako se lahko razlikujejo. Čeprav se je to zdelo kot velik minus se je čez čas izboljšalo saj smo opazili, da se enako funkcije oz. metodo lahko uporabijo v več scenarijih. Seveda to potrebuje veliko »castinga« za dosežek kompatibilnosti tipov. Z tem bi rekel, da je začeten kapital časa visok, a enkrat ko je osnova narejene se lahko z lahkoto prilagodi potrebam saj so podatki kar dostopni.

9 DRUŽBENA ODGOVORNOST

Odgovornost te naloge je po naših ugotovitvah dejstvo, da omogoča prihranek časa in izobraževalne ter izboljševalne možnosti že obstoječim programerjem programskih jezikov JS in TS. Ta prihranjen čas se lahko nato poda k drugim dejavnostim ter s tem poveča produktivnost ali preusmeri nazaj k učenju in raziskovanju željenega ciljnega jezika.

10 ZAKLJUČEK

Program in sam TypeScript, kot jezik imata veliko možnosti za nadaljnji razvoj na vseh področjih. Ker programiranje ni vezano na jezik, okolje ali državo se lahko takšni in podobni projekti razvijajo na globalnem nivoju. Ena izmed takšnih platform je GitHub, ki omogoča enostavno razvijanje ali možnost pomoči mnogim OSS (Open Source Software) projektom kot je tale.

Nameni tega projekti so doseženi čeprav smo morda zašli, a sedaj se lahko še samo izboljšuje in doseže višji nivo.

LITERATURA IN VIRI

LITERATURA

Žan Vidrih, Tomaž Leopold (2015). Interpretacija in pretvorba programskih jezikov. Pridobljeno 01. 02. 2020 s https://zpm-mb.si/wp-content/uploads/2015/06/S%C5%A0_Ra%C4%8Dunalni%C5%A1tvo_Interpretacija_in_pretvorba_programskih_jezikov.pdf

VIRI

Microsoft (2020). TypeScript. Pridobljeno 01. 02. 2020 s https://www.typescriptlang.org/

Microsoft (2020). TypeScript. Pridobljeno 21. 10. 2019 s https://github.com/microsoft/TypeScript/wiki/Using-the-Compiler-API

David Sherret (2020). TypeScript AST Viewer. Pridobljeno 21. 11. 2019 s https://ts-ast-viewer.com/

Wikipedia (2020). Abstract Syntax Tree . Pridobljeno 09. 11. 2019 s https://en.wikipedia.org/wiki/Abstract_syntax_tree