Moon

Co je Moon?

Moon je jednoduchý programovací jazyk vytvořen pro ročníkovou práci. Implementace je provedena v jazyce Rust. Práce vychází z této knihy, kde je jazyk implementován v jazyce Java. Moon je interpretovaný a dynamický, dal by se tedy přirovnat k Pythonu. Kód se nemusí kompilovat a spouští se rovnou. Jazyk umí základní věci jako jsou například jednoduché početní úkony, funkce, třídy, smyčky a cykly.

Jak nainstalovat Moon?

Požadavky pro instalaci (předpokládá se Linux):

- Rust
- git

K instalaci jazyka Moon je potřeba mít nainstalovaný Rust. Rust je jazyk kompilovaný, takže je potřeba kód k Moon zkompilovat. Instalace Rustu je velmi jednoduchá díky stránce rustup. Po spuštění příkazu který se na stránce nachází se Rust stáhne i potřebnými komponenty. Dále je potřeba mít nainstalovaný git. Zdrojový kód se nachází na GitHubu, je tedy potřeba ho z GitHubu stáhnout. Pokud je obojí nainstalováno, je už zprovoznění jazyka Moon jednoduché.

Nejprve zkopírujeme kód z GitHubu do libovolného místa na počítači, stačí mít v tomto místě otevřený terminál a spustit tento příkaz: `git clone https://github.com/sixbytesdeep/moonx`

Poté se pomocí příkazu přesuneme do nově vytvořené složky: `cd moonx`

Teď je potřeba kód zkompilovat. Rust ke kompilaci používá příkaz `cargo`. Kompilaci tedy spustíme příkazem: `cargo b --release`

Písmenko `b` značí zkratku pro "build", v češtine "sestavit". Argument `--release` provede optimalizace pro tzv. "release verzi". Bez tohoto argumentu je sestavena "debug verze", která s sebou nese i informace pro případný debugging a z toho vyplývá, že je pomalejší.

Nyní je potřeba vygenerovaný spustitelný soubor přesunout na místo, kde ho Linux uvidí. Linux se dívá do složek, které jsou v systémové proměnné `PATH`. Složka `/usr/bin` se tam nachází vždy, proto doporučuji přesunout tam. Provedeme příkazem: `sudo cp target/release/moon /usr/bin/moon`.

Nyní už půjde jazyk spustit příkazem `moon`.

Jak Moon používat?

Moon má dva módy ve kterých se dá spustit. Po napsání pouze příkazu `moon`, se spustí interaktivní příkazová řádka, neboli REPL (read-eval-print-loop). Do této příkazové řádky jdou přímo vkládat příkazy, které má jazyk provést. Stejné jako REPL Pythonu.

V druhém módu je potřeba uvést jméno souboru který chceme spustit. To provedeme příkazem: `moon <jmeno souboru>`. Soubory by měly mít koncovku `.lox`.

Syntax je podobný nejblíže JavaScriptu.

Zde si ukážeme pár příkladů:

1. Hello world!

```
> print "Hello World!";
"Hello World"
```

2. Proměnné

```
> var i = 10;
> var jmeno = "alexandr";
> var pravda_nebo_lez = false;

> print i;
10
> print jmeno;
"alexandr"
> print pravda_nebo_lez;
false
```

3. Matematické operace

```
> 5 + 5;
10
> 4 - 3;
1
> 3 * 5;
15
> 6 / 2;
3
```

4. Funkce

```
fun pozdrav(jmeno) {
    print "Ahoj"+jmeno;
};

pozdrav("sasa");

fun returning(cislo) {
    return cislo+5;
}

var n = returning(5);
print n; // zobrazi 10
```

Funkce podporují rekurzi.

5. `if` a `else`

```
var cislo = 10;
if (cislo > 10) {
    print "vetsi";
}
else {
    print "mensi";
}
```

6. `for` a `while`

```
for (var i; i < 10; i = i + 1) {
    print i;
}

var n = 0;
while (n != 10) {
    n = n + 1;
}
```

U `for` loops je momentálně bug u kterého jsem ještě nenašel řešení. Hodnoty proměnné `i` jsou zobrazeny dvakrát.

7. Třídy

```
class Jidlo {
    snist() {
        print "mnam mnam";
    }
}

var jidlo = Jidlo();
jidlo.snist(); // "mnam mnam"
jidlo.navic = "omacka";
print jidlo.navic; // "omacka"

class Zvire {
    jsem() {
        print "Momentalne jsem " + this.jmeno + "!";
    }
}

var zvire = Zvire();
zvire.jmeno = "medved";
zvire.jsem(); // "Momentalne jsem medved!"

class Krabice {
    init(vec) {
        this.vec = vec;
    }
}

var krabice1 = Krabice("vajicka");
print krabice1.vec; // "vajicka"
var krabice2 = Krabice("mrkev");
print krabice2.vec; // "mrkev"
```

8. Prvních 20 čísel Fibonacciho sekvence

```
fun fib(n) {
    if (n <= 1) return n;
    return fib(n-2) + fib(n-1);
}

for (var i = 0; i < 20; i = i+1) {
    print fib(i);
}
```

Zdroje

NYSTROM, Robert. Crafting Interpreters [online]. 2015 - 2021 [cit. 2022-05-18]. Dostupné z: https://craftinginterpreters.com/contents.html

Stack Overflow [online]. [cit. 2022-05-18]. Dostupné z: https://stackoverflow.com/

Rust Book [online]. [cit. 2022-05-18]. Dostupné z: https://doc.rust-lang.org/book/

Praktická část seminární práce se nachází na https://github.com/sixbytesdeep/moonx.