Možnosti využití Pythonu pro tvorbu webových aplikací

Martin Goldbach

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Goldbach**
Osobní číslo:	**A20099**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Možnosti využití Pythonu pro tvorbu webových aplikací**
Téma práce anglicky:	**Possibilities of Using Python for Creating Web Applications**

Zásady pro vypracování

1. Proveďte literární rešerši na zadané téma.
2. Uveďte a popište vlastnosti jazyka Python.
3. Popište možnosti využití jazyka Python na straně webového prohlížeče.
4. Popište vybrané projekty a způsoby řešení, které používají.
5. Seznamte se s technologií PyScript, která umožňuje přímo do webových stránek vkládat kód zapsaný v Pythonu.
6. Na praktických ukázkách uveďte možnosti jeho použití.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. MASTRODOMENICO, Rob. *The Python book*. Hoboken, NJ: John Wiley & Sons, 2022, 1 online resource. ISBN 9781119573364. Dostupné také z: https://proxy.k.utb.cz/login?url=https://onlinelibrary.wiley.com/doi/book/10.1002/9781119573364
2. *PyScript | Run Python in your HTML* [online]. Anaconda, © 2022 [cit. 2022-11-18]. Dostupné z: https://pyscript.net/
3. *WebAssembly* [online]. World Wide Web Consortium, 2022 [cit. 2022-11-18]. Dostupné z: https://webassembly.org/
4. *Pyodide: Version 0.21.3* [online]. Pyodide contributors and Mozilla, © 2019-2022 [cit. 2022-11-18]. Dostupné z: https://pyodide.org/en/stable/
5. TIŠNOVSKÝ, Pavel. PyScript: další technologie umožňující využití Pythonu v prohlížeči. *Root.cz* [online]. Praha: Internet Info, © 1997 – 2022, 20.9.2022 [cit. 2022-11-18]. Dostupné z: https://www.root.cz/clanky/pyscript-dalsi-technologie-umoznujici-vyuziti-pythonu-v-prohlizeci/
6. TIŠNOVSKÝ, Pavel. PyScript: Python ve webovém prohlížeči (dokončení). *Root.cz* [online]. Praha: Internet Info, © 1997 – 2022, 27.9.2022 [cit. 2022-11-18]. Dostupné z: https://www.root.cz/clanky/pyscript-dalsi-technologie-umoznujici-vyuziti-pythonu-v-prohlizeci/

Vedoucí bakalářské práce: **Ing. Petr Navrátil, Ph.D.**
Ústav řízení procesů

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

L.S.

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Martin Goldbach, v. r.
podpis studenta

ABSTRAKT

Bakalářská práce se zabývá možnostmi využití programovacího jazyka Python pro tvorbu webových aplikací na straně prohlížeče. Teoretická část je zaměřena na popis základních vlastností jazyka Python, jeho roli ve webovém vývoji a webovým aplikacím obecně. Dále se práce věnuje popisu vybraných projektů, které takové použití jazyka Python umožňují a způsobu řešení, které využívají. Důraz je kladen na zkoumání a popis projektu PyScript. Praktická část práce se zaměřuje na konkrétní ukázky možností tohoto nástroje.

Klíčová slova: Python, webové aplikace, prohlížeč, PyScript, transpiler, WebAssembly, HTML

ABSTRACT

The bachelor thesis deals with the possibilities of using the Python programming language for creating web applications on the browser side. The theoretical part is focused on the description of the basic features of the Python language, its role in web development and web applications in general. Furthermore, the thesis is devoted to the description of selected projects that allow such use of the Python language and the way they use it. Emphasis is placed on the examination and description of the PyScript project. The practical part of the thesis focuses on specific examples of the capabilities of this tool.

Keywords: Python, web applications, browser, PyScript, transpiler, WebAssembly, HTML

Tímto bych chtěl poděkovat Ing. Petru Navrátilovi, Ph.D. za vedení mé práce, cennou zpětnou vazbu a zodpovězené dotazy.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

V dnešní době se stále více rozvíjí tvorba webových aplikací a mnoho vývojářů hledá efektivní a flexibilní způsob, jak je vytvářet. Jedním z velmi populárních programovacích jazyků, který je vhodný nejen pro tvorbu webových aplikací, je Python. Ačkoliv jeho využití sahá do mnoha oblastí, v případě webových aplikací nachází své místo hlavně na straně serveru, kdy jsou aplikace hostovány na serverech a slouží ke zpracování dat, komunikaci s databázemi a dalším server-side úlohám. Pro tyto účely je k dispozici v Pythonu velké množství knihoven a frameworků, které značně usnadňují vývoj a umožnují vytvářet robustní aplikace s mnoha funkcemi.

Nicméně má v dnešní době Python své místo také na straně prohlížeče neboli klienta, kde může být využíván v kombinaci s dalšími jazyky. Pro tyto účely existují nástroje a knihovny, které umožňují vyvíjet klientskou část webových aplikací v Pythonu a dovolují vývojářům psát Python kód, který běží přímo v prohlížeči.

Cílem práce je prozkoumat tyto dostupné projekty s hlavním důrazem na technologii PyScript, představit řešení, které projekty používají, a demonstrovat praktické použití PyScriptu na konkrétních příkladech.

V teoretické části se práce věnuje programovacímu jazyku Python, jeho vlastnostem, roli ve vývoji webových aplikací a možností jeho využití na straně webových prohlížečů. Dále se práce zaměřuje na způsoby a vybrané projekty, které tuto integraci do prohlížeče umožňují. V praktické části se práce věnuje frameworku PyScript. PyScript umožňuje přímo do webových stránek vkládat kód zapsaný v Pythonu, a tak zjednodušuje integraci tohoto jazyka do webových aplikací. Práce se zaměřuje na praktické ukázky využití tohoto frameworku.

Výsledkem bakalářské práce by měl být ucelený přehled, přispívající k rozšíření vědomostí o současných možnostech využití Pythonu v oblasti webových aplikací a praktické ukázky použití technologie PyScript.

I. TEORETICKÁ ČÁST

1 PYTHON

1.1 Úvod do jazyka Python

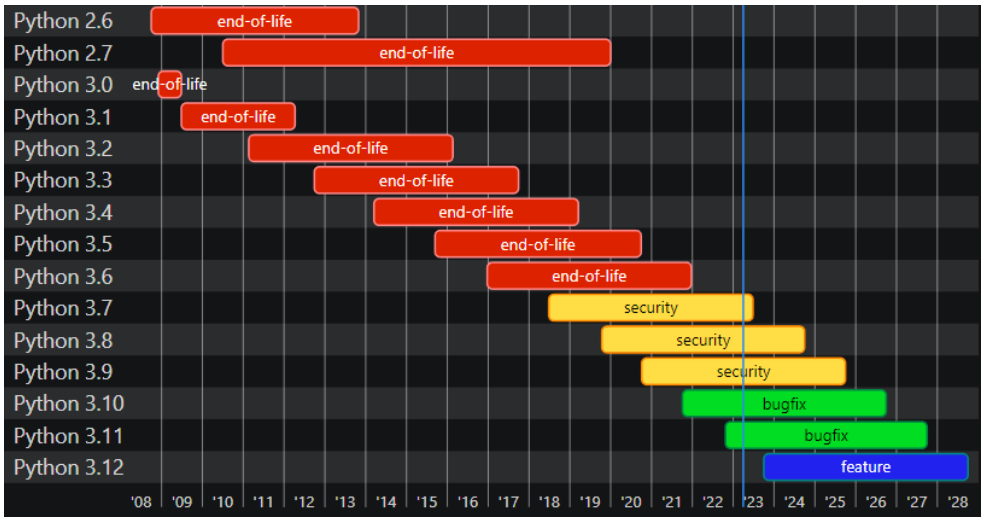
Python lze označit za moderní, vysokoúrovňový, interpretovaný programovací jazyk. Kód tedy nemusí být před spuštěním zkompilován kompilátorem, ale je překládán za běhu pomocí interpreteru do strojového kódu. Vyznačuje se svou jednoduchostí, flexibilitou, a hlavně širokým využitím v mnoha různých oblastech. [1]

Obrázek 1 - Logo Pythonu [2]

1.1.1 Historie a vývoj

Python je jazyk s bohatou historií. Jeho vývoj začal v roce 1989, když Guido van Rossum pracoval na nápadu jednoduššího a snadno čitelného skriptovacího jazyka inspirovaného jazykem ABC. Název Python byl zvolen Guidem van Rossumem, jako odkaz na britskou komediální skupinu Monty Python. [3]

První verze Pythonu 0.9.0 byla zveřejněna v roce 1991. Následovaly další významné verze, jako Python 1.0 (1994), Python 2.0 (2000) a Python 3.0 (2008). Python 3 byl zásadním krokem vpřed, i když způsobil některé komplikace kvůli zpětné nekompatibilitě. [3] Aktuálně nejnovější stabilní verze Pythonu je Python 3.11, poprvé vydána v roce 2022. [4]



Obrázek 2 - Cyklus vývoje verzí Pythonu [4]

Na předchozím obrázku lze vidět cyklus vydání jednotlivých verzí Pythonu od verze 2.6 až po aktuálně nejnovější vyvíjenou verzi 3.12.

Tyto stavy vývoje popisují různé fáze životního cyklu verzí Pythonu, které zahrnují přidávání nových funkcí, opravy chyb a zabezpečení, až po konečné ukončení podpory dané verze. [4]

1.2 Syntaxe

Velmi opěvovanou vlastností Pythonu je jeho syntaxe. Právě díky jednoduchosti je Python často první volbou pro začínající programátory, výuku programování a rychlý vývoj. [1] Jednoduchá a čitelná syntaxe usnadňuje učení a zvyšuje produktivitu vývojářů, což z něj činí oblíbený jazyk v mnoha průmyslových odvětvích.

1.2.1 Čitelnost

Jazyk je navržen tak, aby byl co nejméně náročný na psaní a zároveň snadno srozumitelný. Důležitou roli v syntaxi jazyka hraje odsazení. Oproti jiným programovacím jazykům, u kterých je odsazení kódu především pro lepší čitelnost, v Pythonu odsazení značí blok kódu. Blok programu tedy neumisťujeme do složených závorek. [5] Ukončení jakéhokoliv příkazu se provádí vložením nového řádku. V jiných programovacích jazycích to zpravidla bývá pomocí středníku (;). Na ukázce níže je vidět způsob zápisu definice funkce a následně její volání.

```
def funkce(parametr):
    if parametr == "Python":
        print ("Ahoj světe")

funkce("Python")
```

1.2.2 Datové typy

Jedná se o dynamicky typovaný programovací jazyk, což znamená, že proměnné nemusí být explicitně deklarovány s určitým datovým typem. Datový typ proměnné je automaticky určen při přiřazení hodnoty a může se měnit během provádění programu.

Vzhledem k tomu, že Python je objektově orientovaný jazyk, vše je reprezentováno jako objekt. V případě datových typů můžeme říct, že datový typ je třídou, a proměnná konkrétního datového typu je instancí této třídy. Součástí každé z těchto tříd jsou metody a vlastnosti, které slouží k manipulaci s daty daného datového typu. [1]

Mezi základní vestavěné datové typy patří:

- číselné typy:
  - *int* – celá čísla
  - *float* – desetinná čísla
  - *complex* – komplexní čísla
- textové typy:
  - *str* – textový řetězec
- booleovské typy:
  - *True*
  - *False*
- datové typy pro uchovávání dat:
  - *list* – kolekce dat
  - *tuple* – neměnitelná kolekce dat
  - *dict* – slovník, data uložena v páru formou „klíč-hodnota“
  - *set* – kolekce dat, nemůže obsahovat duplicitní hodnoty [1]

1.3 Knihovny a moduly

1.3.1 Standartní knihovna

Python Standart Library neboli standartní knihovna je rozsáhlá knihovna funkcí a modulů, které jsou distribuovány s jazykem Python. Obsahuje jak vestavěné moduly, které poskytují přístup k systémovým funkcím, tak i moduly napsané v samotném Pythonu, které nabízejí standardizovaná řešení pro běžné problémy v programování. Mezi tyto moduly patří například moduly pro práci s matematickými operacemi a čísly (math, random), moduly pro přístup k souborům a složkám (fileinput, os.path), moduly pro komprimaci dat (zlib, bz2) a mnoho dalších. [6]

1.3.2 Externí knihovny

Kromě standardní knihovny existují tisíce dalších knihoven a modulů dostupných prostřednictvím Python Package Index (PyPI), které umožňují rychle a snadno implementovat různé funkce. Tyto knihovny zahrnují nástroje pro webový vývoj (jako Django a Flask), datovou analýzu (jako NumPy a pandas) a strojové učení (jako scikit-learn a TensorFlow). [6; 7]

1.4 Implementace jazyka Python

Existuje několik implementací Pythonu, které poskytují různé funkce, vlastnosti a zaměřují se na určité oblasti a platformy. Několik těchto implementací je představeno níže.

1.4.1 CPython

CPython je standardní a nejpoužívanější implementace Pythonu. Je napsaný v jazyce C a je referenční implementací jazyka Python. Jedná se tedy o implementaci Pythonu, která je dostupná na oficiálním webu python.org.

1.4.2 Jython

Jython je implementace Pythonu, která je kompatibilní s platformou Java. Jython překládá Python kód do bytecode Java Virtual Machine (JVM), což umožňuje integraci s existujícím Java kódem a knihovnami. [8]

1.4.3 IronPython

IronPython je implementace Pythonu, která je navržena pro běh na platformě .NET. Je kompatibilní s jazyky a knihovnami .NET Frameworku a umožňuje integraci s existujícím .NET kódem. IronPython poskytuje možnost využití Pythonu pro vývoj aplikací v prostředí .NET. [9]

1.4.4 PyPy

PyPy je alternativní implementace Pythonu, která je postavena na RPython (Restricted Python) platformě. PyPy se zaměřuje na výkon a využívá Just-in-Time (JIT) kompilátor, který umožňuje rychlejší provádění kódu ve srovnání s klasickým CPythonem. PyPy je kompatibilní s většinou kódu psaného pro CPython a podporuje několik verzí Pythonu. [10]

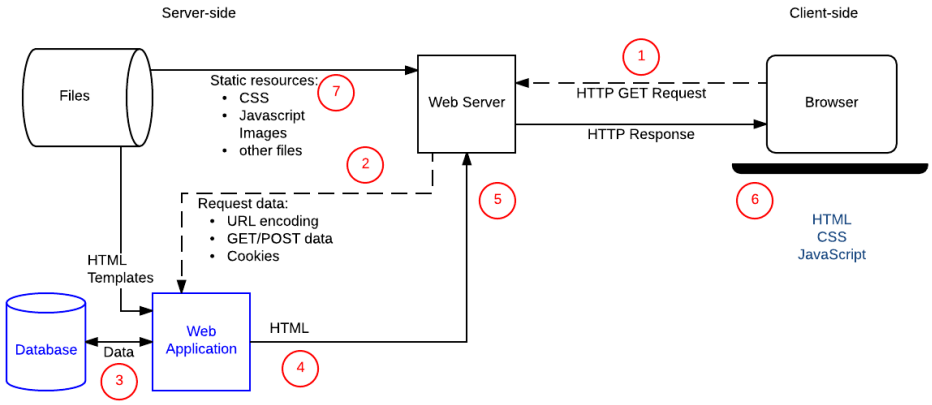
2 WEBOVÉ APLIKACE

2.1 Úvod do webových aplikací

Webové aplikace jsou software navržený k běhu v internetovém prohlížeči a slouží k interakci s uživatelem prostřednictvím webových stránek. Tento druh aplikací se stal základním kamenem dnešního internetu, protože umožňuje uživateli přistupovat k funkcím a službám přímo z prohlížeče, bez nutnosti instalace samotného softwaru. [11]

2.2 Architektura webových aplikací

Webové aplikace obvykle využívají tzv. klient-server architekturu. Klientem se rozumí webový prohlížeč, který komunikuje se serverem. Klient požádá server o zaslání webové stránky, server žádost zpracuje, popřípadě ještě získá data z databáze a vrátí odpověď ve formě HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) a JavaScriptu, které prohlížeč zobrazí jako webovou stránku. [12]



Obrázek 3 - Klient-server architektura [12]

V oblasti vývoje lze tedy webovou aplikaci dělit do dvou hlavních částí:

- Frontend (client-side) – týká se uživatelského rozhraní, tedy to, co vidí uživatel a děje se na straně prohlížeče. Zahrnuje technologie jako HTML, CSS a JavaScript.
- Backend (server-side) – zabývá se děním na straně serveru, zpracováním dat, logikou aplikace a komunikací se serverem, databázemi nebo jinými službami.

Každá z těchto částí využívá specifické jazyky, jejich frameworky a technologie, které jsou pro danou oblast nejvhodnější a společně se podílejí na vytváření komplexních webových aplikací.

2.3 Technologie webových aplikací na straně prohlížeče

2.3.1 HTML

HTML (HyperText Markup Language) je značkovací jazyk, který popisuje strukturu a definuje význam obsahu webové stránky. Pojem *HyperText* označuje propojení mezi jednotlivými webovými stránkami pomocí odkazů. Tyto odkazy mohou spojovat jednotlivé části textu nebo celé webové stránky, a to jak v rámci jednoho webu, tak napříč různými weby. Slovo *Markup* v názvu odkazuje na tzv. tagy, které se používají k označení samotného obsahu a definují tak jeho význam. Názvy jednotlivých tagů se zapisují společně s jejich atributy, které umožňují měnit jejich vlastnosti, do závorek „<“ a „>“. [13]

Obrázek 4 – Ukázka HTML elementu [13]

Na obrázku výše je ukázka * * elementu, který reprezentuje odstavec. Obsah (text) je obalen pomocí * * tagů společně s atributem *class*, který slouží k rozlišení elementu pro jeho pozdější manipulaci.

2.3.2 CSS

Cascading Style Sheets (CSS) je označení pro kaskádové styly, které definují, jakým způsobem budou jednotlivé HTML elementy zobrazeny. To nám umožňuje popsat celkový vzhled webové stránky. CSS je jazyk se svou vlastní syntaxí. Samotné stylování elementu probíhá výběrem konkrétního elementu z HTML souboru (tzv. selektor) a následně definováním jeho vlastností (např. barva, velikost, pozice atd.). Pomocí CSS lze dosáhnout přesné kontroly nad elementy HTML stránky, rozložením a dalšími vizuálními prvky, což zjednodušuje úpravy a zajišťuje konzistentnost formátování na celém webu. [14]

2.3.3 JavaScript

JavaScript je důležitým a mnohostranným programovacím jazykem používaným především ve webovém vývoji, který umožňuje tvorbu dynamických a interaktivních webových stránek a aplikací. Jazyk běží na straně klienta, což znamená, že veškeré zpracování probíhá v prohlížeči uživatele. Díky JavaScriptu je možné měnit obsah webové stránky, reagovat na vstup od uživatele, vytvářet dynamická menu, validovat webové formuláře a vytvářet různé

vizuální efekty. Mnohostrannost tohoto jazyka dokazuje také to, že je jej možné využít i na straně serveru. A to díky prostředí Node.js [15]

V současnosti obvykle probíhá vývoj webových aplikací za pomoci některých JavaScript frameworků, které poskytují předem vytvořené funkce, komponenty, struktury a usnadňují tak vývojářům práci. Mezi tyto frameworky patří např. React, vytvořen v roce 2013 společností Facebook, Angular, postaven na TypeScriptu, framework Vue.js a další. [16]

2.4 Python a jeho role ve webových aplikacích

Jak již bylo zmíněno, v oblasti webového vývoje je jazyk Python situován hlavně na straně serveru a není tedy primárně používanou technologií v klientské části webových aplikací.

2.4.1 Webové frameworky

V rámci backend vývoje nachází Python své uplatnění v odesílání a zpracování dat, komunikací s databázemi či jinými servery, směrování URL, zajištění bezpečnosti a další. K těmto účelům a pro zjednodušení procesu vývoje, existují Python frameworky, které poskytují hotová řešení a nástroje pro vývoj webových aplikací.

Mezi tyto velmi populární webové frameworky paří například Django, Flask, CherryPy, Web2Py a další. [17]

2.4.2 Python a známé webové aplikace

Netflix, přední streamovací platforma, stojí na jádru založeném na Pythonu. Python se zde využívá pro širokou škálu aplikací, včetně nástrojů pro zabezpečení, systému pro doporučení obsahu a obsahové distribuční sítě Open Connect, která zajišťuje plynulý přenos videí pro uživatele Netflixu. [18]

Další populární webovou platformou pro sdílení příspěvků a diskusí, využívající jazyk Python, je Reddit. Python zde funguje jako prostředník mezi uživatelskými požadavky a zobrazovanými informacemi. [18]

Amazon, světový gigant v oblasti e-commerce, inovativně využívá Python k vylepšení uživatelského zážitku. Použití algoritmů pro strojové učení v Pythonu umožňuje Amazonu analyzovat data a doporučovat uživatelům personalizované produkty. [18]

3 PYTHON NA STRANĚ PROHLÍŽEČE

3.1 Úvod do problematiky

V oblasti frontendu je JavaScript hlavním, webovými prohlížeči nativně podporovaným programovacím jazykem pro vývoj interaktivních webových aplikací. Je tomu tak hlavně z historických důvodů. Psaní kódu s využitím jiných programovacích jazyků tedy může být trochu komplikovanější a žádá si tak použití jiných metod. [19]

Historie JavaScriptu sahá až do roku 1995. V této době měly webové stránky pouze statický obsah. V důsledku toho byla snaha toto omezení napravit. Společnost Netscape vytvořila v roce 1995 JavaScript jako doprovodný jazyk k Javě pro jejich webový prohlížeč Navigator. Cílem bylo udělat z webu plnohodnotnou platformu pro aplikace, přičemž JavaScript by běžel jak na klientovi, tak na serveru. Zatímco na backendu trvalo mnoho let, než byl JavaScript brán vážně, na frontendu se rychle stal standardním programovacím jazykem. Velkou zásluhu na rozšíření si také připisuje společnost Google, která ve svém Gmailu využila technologii AJAX, založenou na JavaScriptu. Tato technologie umožňuje načíst části webu bez nutnosti opakovaného načítání celé stránky. K rychlému vzestupu také přispělo předložení JavaScriptu společnosti ECMA (European Computer Manufactures Association) pro standardizaci a zajištění řádné údržby a podpory tohoto jazyka. [20; 15]

I přes takto dominantní postavení jazyka JavaScript je snaha o umožnění vývojářům používat v prohlížeči jiný jazyk a využívat tak jeho předností. Proto vznikají projekty a nástroje, které se snaží toto omezení vyřešit a dovolit tak použití jiných programovacích jazyků. V následující části se seznámíme s řešeními, na kterých jsou právě tyto nástroje a projekty postaveny.

3.2 Transpiler – překladač zdrojového kódu

3.2.1 Překladače

Obecně tak můžeme nazvat programy, které překládají zdrojový kód do jiného typu kódu. Rozlišujeme dva základní typy překladačů:

- Interpreter – překládá zdrojový kód, přičemž je zároveň vykonáván,
- Kompilátor – překládá kód vysokoúrovňových jazyků na strojový kód a uloží spustitelní kód do paměti. Tento kód je poté možné spustit a je schopen běžet na procesoru. [21]

3.2.2 Co je to transpiler?

Transpiler (také *source-to-source compiler, transcompiler, source-to-source translator*) je druh překladače, který převádí zdrojový kód napsaný v jednom jazyce (zdrojový jazyk) do ekvivalentního zdrojového kódu v jazyce jiném (cílový jazyk). Transpiler je podobný kompilátoru, který převádí zdrojový kód jazyka s vysokou úrovní abstrakce do strojového kódu nebo jazyka s úrovní nižší, než je on sám. Příkladem může být kompilace jazyka C do strojového kódu. Oproti tomu transpiler, umožňuje překlad zdrojového kódu jazyka, který je na stejné úrovni abstrakce jako jazyk cílový. [22]

Jedním z běžně známých transpilerů je např. Babel. Tento nástroj umí převádět moderní JavaScript kód (např. napsaný v ES6) na starší verze JavaScriptu, aby byl kompatibilní s prohlížeči a platformami, které ještě nepodporují nejnovější verze jazyka. [23]

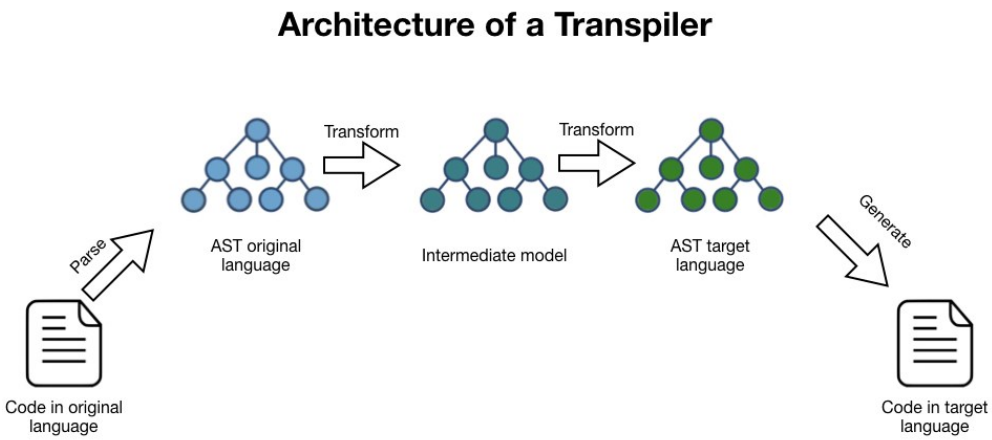
Překlady mezi jednotlivými verzemi Javascriptu to samozřejmě nekončí. K dispozici existuje spousta takových nástrojů, včetně Python transpilerů, kterým se podrobněji budeme věnovat později. Seznam těchto dostupných projektů lze najít na následující adrese (https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS). [24]

Je tedy jasné, že takový překlad do jiných jazyků umožňuje vývojářům psát kód v jazyce, který je pro ně pohodlnější, vhodnější a poté jej převést do jazyka, který je kompatibilní s cílovou platformou.

3.2.3 Jak to funguje?

Transpilaci zdrojového kódu můžeme rozdělit do tří základních kroků. Jako první proběhne tzv. „*parsování*“ kódu. Cílem parsování je za pomoci lexikální a syntaktické analýzy vytvoření abstraktního syntaktického stromu (AST). Lexikální analyzátor rozdělí vstupní zdrojový kód na jednotlivé výrazy (proměnné, klíčová slova, operátory atd.), zvané tokeny. Tomuto procesu se také říká tokenizace. Za pomoci Syntaktické analýzy jsou tyto tokeny převedeny do již zmíněné stromové struktury AST. Abstrakční syntaktický strom nám tak reprezentuje jednotlivé části původní syntaxe a vztahy mezi nimi. Následuje fáze transformace. Již vytvořený abstrakční strom je transformován do abstrakčního stromu cílového programovacího jazyka. Při transformaci dochází ke změnám nebo i odstranění jednotlivých uzlů stromu. Složitost tohoto kroku je ovlivněna podobností zdrojového a cílového programovacího jazyka. Posledním krokem transpilace je generování. V tomto kroku je z modifikovaného abstraktního syntaktického stromu generován zdrojový kód cílového jazyka. [25; 26]

Na obrázku níže je vyobrazena architektura transpileru.



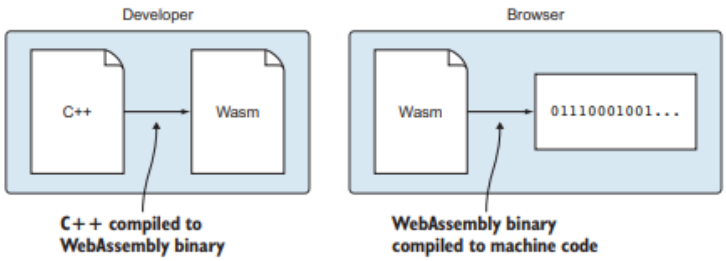
Obrázek 5 – Architektura transpileru [22]

3.3 WebAssembly

Dle oficiálního webu je WebAssembly definován následovně:

„WebAssembly (abbreviated Wasm) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.“ [27]

Jedná se tedy o jazyk podobný assembleru, který byl navržen tak, aby poskytoval kompaktní binární soubory, které mohou být rychle přeneseny, staženy a následně spuštěny webovými prohlížeči v téměř nativní rychlosti. WebAssembly slouží jako cílový formát kompilace vysokoúrovňových programovacích jazyků a není tedy určen, aby jeho byl kód psán ručně. Tento formát poté může běžet např. ve webovém prohlížeči, který podporuje WebAssembly, a je překládán do nativního strojového jazyka. Důležité je také zmínit, že cílem WebAssembly není nahradit jazyk JavaScript, nýbrž jít ruku v ruce s ním a poskytovat efektivnější prostředky pro výkonově náročné části webových aplikací. [28]



Obrázek 6 – Kompilace C++ do Wasm a následně Wasm do strojového kódu [28]

3.3.1 WebAssembly jako webový standard

WebAssembly byl poprvé oznámen v roce 2015. V té době prohlížeče podporovaly tři jazyky (HTML, CSS, JavaScript). Během následující let však velké společnosti jako Google (Chrome), Microsoft (Edge), Apple (Safari) postupně zařazují podporu WebAssembly do svých prohlížečů. [29] V prosinci 2019, světové konsorcium (W3C) oficiálně uznalo WebAssembly jako webový standard, a tím se stává čtvrtým podporovaným jazykem prohlížečů. [30]

V současné době, dle statistik z webu caniuse.com, je WebAssembly kompatibilní s více než 95 % prohlížeči včetně těch pro mobilní zařízení. [31]

3.3.2 Co řeší WebAssembly?

Hlavní problém, který se formát WebAssembly (Wasm) snaží vyřešit je rychlost. JavaScript je interpretovaný programovací jazyk, což znamená, že musí být přeložen na strojový kód při každém spuštění kódu. To může vést k pomalejšímu provádění kódu a omezeným optimalizacím. Oproti tvorbě celé webové aplikace v JavaScriptu lze díky WebAssembly psát kód v různých programovacích jazycích, jako C, C++, Rust a další. Tento kód je poté kompilován do formátu WebAssembly bajtkódu, což umožňuje spouštění aplikací náročných na výkon přímo v prohlížeči bez nutnosti stahovat několika megabajtové JavaScript soubory. Když se kód jednou zkompiluje do WebAssembly, tak už nemusí být znovu kompilován. Mimo to, že se jedná o binární formát, který je mnohem menší než textová forma JavaScriptu, umožňuje tedy WebAssembly kompilaci na nižší úrovni bez nutnosti opakované interpretace jazyka. [28]

3.3.3 Bezpečnost WebAssembly

WebAssembly poskytuje izolované prostředí pro běh aplikací v prohlížeči. Formát WebAssembly ve webovém prohlížeči sdílí stejné běhové prostředí jako JavaScript (JavaScript Virtual Machine). Moduly WebAssembly tak nemají přístup k žádným prostředkům, ke kterým by neměl přístup i JavaScript. [28]

3.4 Řešení pro jazyk Python

V této části se budeme věnovat, jakým způsobem je možné jazyk Python integrovat a spouštět na straně webového prohlížeče. Představíme si tedy konkrétní projekty a nástroje, které toto umožňují a vysvětlíme si metody které používají.

3.5 Brython

Brython je interpret jazyka Python napsaný v jazyce JavaScript. Název "Brython" je kombinací slov "Browser" a "Python". Jedná se o transpiler z jazyka Python do JavaScriptu, který umožňuje spouštět kód Pythonu přímo ve webovém prohlížeči a samotná transpilace probíhá na pozadí při načítání stránky. Hlavním cílem projektu Brython, ostatně jako u většiny těchto projektů, je nahradit JavaScript Pythonem, jako skriptovacího jazyka pro webové prohlížeče. [32]

3.5.1 Použití nástroje Brython

Pro lokální instalaci Brythonu je možné použít pip (*pip install brython)*. Alternativní a jednodušší řešení je přidání Brythonu přímo do HTML souboru pomocí CDN (Content Delivery Network). Důležité je, abychom měli přístup k souborům *brython.js* a *brython_stdlib.js*. Tyto soubory poté připojíme k HTML souboru pomocí <*script*> tagu. Soubor *brython.js* obsahuje základní engine Brythonu a nejčastěji využívané moduly pro práci s HTML elementy (modul *browser*). Soubor *brython_stdlib.js* seskupuje některé moduly a balíčky standartní knihovny Python. [32]

```
<> index.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3
4   <head>
5       <meta charset="UTF-8">
6       <meta http-equiv="X-UA-Compatible" content="IE=edge">
7       <meta name="viewport" content="width=device-width, initial-scale=1.0">
8       <title>Document</title>
9       <script src="https://cdn.jsdelivr.net/npm/brython@3/brython.min.js">
10      </script>
11      <script src="https://cdn.jsdelivr.net/npm/brython@3/brython_stdlib.js">
12      </script>
13  </head>
14
15  <body onload="brython()">
16      <script type="text/python">
17          <!-- Python code -->
18      </script>
19  </body>
20  </html>
```

Obrázek 7 – Integrace Brythonu do HTML souboru

Samotný kód Pythonu se zapisuje do <*script*> tagu, případně do samostatného Python modulu. Ve <script> tagu je potřeba uvést atribut *type* (viz Obrázek 7). Nakonec je nutné zmínit funkci *brython()* v atributu *onload* v body elementu. Tato funkce při načítání webu

vyhledává veškeré <*script*> tagy, obsahující atribut *type="text/python"* a následně přeloží Python kód do JavaScriptu. Ten je poté spouštěn *eval()* funkcí v JavaScriptu. [32]

3.6 Transcrypt

Transcrypt je opět transpiler, který překládá Python kód do jazyka JavaScript. V současnosti podporuje verzi Pythonu 3.9. Transcrypt upřednostňuje především přístup k webově orientovaným JavaScript knihovnám, oproti těm dostupným v Pythonu. Mezi podporované Python knihovny, které lze použít, patří math, cmath, random, itertools, re, time, datetime a turtle. Součástí je také knihovna Numscrypt. Je založena na populární knihovně NumPy, která je používána pro vědecké výpočty a analýzu dat v Pythonu. [33]

Překlad Pythonu probíhá tzv. ahead-of-time (s předstihem). To znamená, že nejprve se musí provést překlad samotného Python kódu, tento překlad vygeneruje JavaScript soubor, a ten je možné vložit pomocí <*script*> tagu do HTML souboru. Tímto se Transcrypt snaží docílit co nejvyšší rychlosti načtení stránek. [34] V tomto ohledu se liší od nástrojů, které překládají kód Pythonu za běhu.

In hello.html, Python handlers are attached directly to the onclick events

```
<script type="module">import * as hello from './__target__/hello.js'; window.hello = hello;</script>
<h2>Hello demo</h2>

<p>
<div id = "greet">...</div>
<button onclick="hello.solarSystem.greet ()">Click me repeatedly!</button>

<p>
<div id = "explain">...</div>
<button onclick="hello.solarSystem.explain ()">And click me repeatedly too!</button>
```

Obrázek 8 – Příklad importu přeloženého skriptu do HTML souboru [35]

3.7 Skulpt

Pomocí nástroje Skulpt lze kód v Pythonu přeložit do JavaScriptu. Jde tedy opět o transpiler, který kód překládá za běhu.

Skulpt je vhodný především pro tvorbu interaktivních programovacích prostředí pro psaní kódu v Pythonu. Příkladem takového prostředí může být projekt Trinket.io. [36] Tato prostředí mohou sloužit jako výukové nástroje nebo částečně nahradit vývojové prostředí či editor.

V současné době se jedná o implementaci Pythonu verze 2.x. v JavaScriptu. Avšak dle dostupných informací v repositáři projektu je kompatibilita s Pythonem verze 3 velkou prioritou a pro většinu použití je již k dispozici. [37]

3.7.1 Použití nástroje Skulpt

Potřebné soubory pro použití nástroje Skulpt lze získat pomocí CND nebo stáhnutím na oficiálním webu projektu. Tyto soubory je nutné opět integrovat do HTML souboru ve *<script>* tagu. Jedná se o soubory *skulpt.js* popřípadě jeho zmenšená verze *skulpt.min.js* a soubor *skulp-stdlib.js* s implementací modulů a funkcí standartní knihovny Pythonu v Javascriptu. [38]

```
3   <script src="skulpt/skulpt.min.js" type="text/javascript"></script>
4   <script src="skulpt/skulpt-stdlib.js" type="text/javascript"></script>
```

Obrázek 9 – Přidání souborů *skulpt.min.js* a *skulpt-stdlib.js* do HTML souboru

Pro spuštění Pythonu v prohlížeči je nezbytné použití JavaScript kódu, který slouží k zachycení kódu napsaného v Pythonu a volání potřebných metod pro jeho syntaktické zpracování a následné vykonání. Příkladem takové funkce v JavaScriptu, která získá kód z textového pole *<textarea>*, umístěné v HTML souboru, je na Obrázek 10 . Python kód (konkrétně obsah v textovém poli) je uložen v proměnné *prog*. Ten je poté přeložen a spuštěn pomocí metody *importMainBody* objektu *Sk* a výstup zobrazen v *<pre>* elementu.

```
// Here's everything you need to run a python program in skulpt
// grab the code from your textarea
// get a reference to your pre element for output
// configure the output function
// call Sk.importMainWithBody()
function runit() {
   var prog = document.getElementById("yourcode").value;
   var mypre = document.getElementById("output");
   mypre.innerHTML = '';
   Sk.pre = "output";
   Sk.configure({output:outf, read:builtinRead});
   (Sk.TurtleGraphics || (Sk.TurtleGraphics = {})).target = 'mycanvas';
   var myPromise = Sk.misceval.asyncToPromise(function() {
       return Sk.importMainWithBody("<stdin>", false, prog, true);
   });
   myPromise.then(function(mod) {
       console.log('success');
   },
       function(err) {
       console.log(err.toString());
   });
}
```

Obrázek 10 – Funkce *runit()* pro provedení kódu v Pythonu [39]

3.8 Pypy.js

V případě Pypy.js se nejedná zcela o transpiler, nýbrž o projekt, který je založen na implementaci jazyka Python s názvem Pypy. Konkrétně jde o JavaScriptovou verzi interpreteru implementace Pypy. Jednou z nevýhod použití PyPy.js na webové stránce je rychlost. PyPy.js obsahuje kompletní interpreter Pythonu, což znamená, že velikost je v řádech megabajtů. To může mít negativní dopad na dobu načítání stránky, zejména při použití CDN a pomalejším internetovém připojení. Pypy.js poskytuje metody pro spouštění Python kódu a předávání proměnných a dat mezi JavaScriptem a Pythonem. Kódu psanému v JavaScriptu se tedy úplně nevyhneme. [36]

3.8.1 Použití nástroje Pypy.js

K použití interpreteru Pypy.js je potřeba do HTML souboru importovat ve *<script>* tagu soubor *pypyjs.py*, *FunctionPromise.js* a *Promise.min.js*. Tím získáme přístup k objektu *pypyjs*, který představuje samotný interpreter v prohlížeči. K manipulaci s intepreterem jsou tady tři důležité metody objektu *pypyjs*. Metoda *exec()* umožňuje vykonat kód v Pythonu, který je předáván parametrem. Dále jsou to metody *set()*, která předává proměnnou z JavaScriptu do Pythonu, a *get()*, která proměnnou získá z Pythonu do JavaScriptu. [40; 36]

Na Obrázek 11 je ukázka použití interpreteru pro vykonání jednoduchého výrazu v Pythonu. Výsledek výrazu je uložen v proměnné *y*, která je předána metodou *pypyjs.get()* a výsledek zobrazen v *alert* okně webové stránky.

```
<body>
    <script type="text/javascript">
        pypyjs.exec(
          // Run some Python
          'y = [x**2. for x in range(10)]'
        ).then(function() {
          // Transfer the value of y from Python to JavaScript
          return pypyjs.get('y');
        }).then(function(result) {
          // Display an alert box with the value of y in it
          alert(result);
        });
    </script>
</body>
```

Obrázek 11 – Použití interpreteru Pypy.js pro vykonání výrazu v Pythonu [36]

3.9 Pyodide

Projekt Pyodide je jedna z mnoha distribucí jazyka Python, která je určena pro webové prohlížeče, založena na překladu do formátu WebAssembly za využití projektu Emscripten (nástroj pro překlad kódu v jazyku C a C++ do WebAssembly). Autorem projektu je Michael Droettboom, který v roce 2018, jako člen společnosti Mozilla, vytvořil Pyodide v rámci projektu Iodide. [41]

Díky překladu implementace jazyka CPython 3.11 [42] (referenční implementace jazyka Python) do formátu WebAssembly je Pyodide schopný spustit Python kód s vysokým výkonem přímo v prohlížeči. Tím odpadá potřeba posílat výpočty na vzdálený server, a umožňuje tak provádět analýzu dat a vizualizaci v reálném čase bez zbytečného přenosu dat. [36]

3.9.1 Podpora balíčků a knihoven

Obrovským přínosem projektu Pyodide je podpora pro populární vědecké knihovny, které jsou základem pro práci s daty a vědeckými výpočty. Mezi tyto knihovny patří NumPy, Matplotlib, SciPy, Pandas a další. Krom toho je samozřejmostí podpora většiny modulů standartní knihovny Python (vyjma modulů curses, dbm, ensurepip, idlelib, lib2to3, tkinter, turtle.py, turtledemo, venv, pwd). Další balíčky je možné v Pyodide instalovat z PyPI pomocí *micropip*. Podmínkou pro instalaci těchto balíčků je, že musí být napsány čistě v Pythonu a musí obsahovat wheel formát [41]. V příloze P I (Tabulka 1) jsou uvedeny názvy všech aktuálně dostupných balíčků třetích stran, které jsou součástí Pyodide.

3.9.2 Použití Pyodide

Pro spuštění Pythonu v prohlížeči pomocí nástroje Pyodide je potřeba získat přístup k souboru *pyodide.js* pomocí CDN. K inicializaci Pyodide poté slouží asynchronní funkce *loadPyodide()*. Spuštění Python kódu provádí metoda *runPython()*, která jako vstupní parametr obdrží textový řetězec kódu v Pythonu. Pro import a použití knihoven a modulů třetích stran, které jsou součástí projektu Pyodide (viz Tabulka 1 v příloze P I), je nutné knihovny načíst metodou *pyodide.loadPackage('název')*. U knihoven a modulů standartní knihovny Python stačí použít import v rámci kódu v Pythonu. [43]

```
<!doctype html>
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/pyodide/v0.23.2/full/pyodide.js"></script>
  </head>
  <body>
    Pyodide test page <br>
    Open your browser console to see Pyodide output
    <script type="text/javascript">
      async function main(){
        let pyodide = await loadPyodide();
        console.log(pyodide.runPython(`
            import sys
            sys.version
        `));
        pyodide.runPython("print(1 + 2)");
      }
      main();
    </script>
  </body>
</html>
```

Obrázek 12 – Příklad použití Pyodide v HTML souboru [43]

3.10 Možné problémy těchto nástrojů

Ačkoliv všechny výše uvedené nástroje, které umožňují Pythonu běžet v rámci webového prohlížeče, svou funkci plní, setkávají se řadou problémů, které mohou omezit jejich použitelnost. Jedním z problémů je podpora knihoven. Ne všechny tyto nástroje podporují knihovny, které jsou často základem aplikací v Pythonu a dovolí tak naplno využít jeho potenciál. Pravděpodobně s nejlepším řešením přichází projekt Pyodide, jehož součástí je podpora těch nejpoužívanějších knihoven v Pythonu.

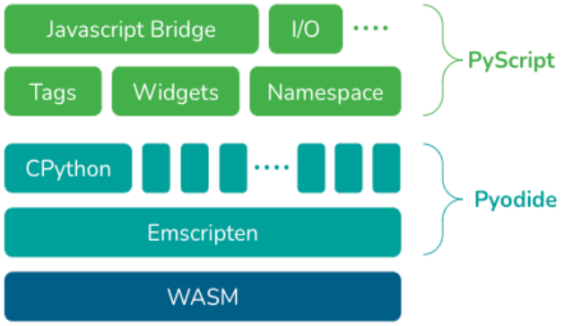
Dalším problémem může být zastaralost. Tyto nástroje často pracují s určitou verzí Pythonu, která může být ve srovnání s nejnovějšími verzemi zastaralá. To znamená, že některé funkce a vylepšení dostupné v novějších verzích Pythonu nebudou k dispozici.

Je nutné zmínit i dostupnost informačních zdrojů pro tyto projekty. Kvůli menšímu rozšíření a použití těchto projektů, oproti JavaScriptu, je také omezený počet zdrojů informací, tutoriálů a podpory dostupné online.

Rychlost je dalším aspektem, který je třeba zvážit při používání těchto nástrojů. Je důležité si uvědomit, že spouštění Pythonu v prohlížeči přináší určitou režii a může být pomalejší než spouštění Pythonu na serveru nebo ve standardním prostředí. To je způsobeno překladem Python kódu do JavaScriptu či kompilací WebAssembly formátu pro běh v prohlížeči.

4 PYSCRIPT

Projekt PyScript je poměrně novou a stále vznikající technologií, kterou představila společnost Anaconda na konferenci PyCon US 2022. Základním kamenem technologie PyScript je projekt Pyodide, který, jak již bylo zmíněno dříve, je přeložená referenční implementace CPythonu do formátu WebAssembly a ten může být spouštěn prohlížečem. [44]



Obrázek 13 – Schéma nástroje PyScript [44]

Cílem PyScriptu je poskytnout nástroj, který jednoduchým způsobem umožňuje použít jazyk Python ve webovém prostředí. Tohoto cíle dosahuje například tak, že nevyžaduje žádnou instalaci a zároveň poskytuje velmi chytré řešení pro vkládání kódu v Pythonu pomocí speciálních elementů a tagů. [45]

Oproti některým projektům zmíněným dříve, hlavním úkolem PyScriptu není nahradit JavaScript. Autoři věří, že PyScript umožní jazyku Python využívat konvence HTML, CSS a JavaScriptu k vytváření uživatelských rozhraní a k řešení problémů při tvorbě webových aplikací. V některých případech bude jeho použití i vhodnější oproti JavaScriptu a zároveň zpřístupní webové programování spoustu lidem, kteří JavaScript nepoužívají. [45]

V současnosti je PyScript stále zatím ve fázi alfa vývoje a sami vývojáři nedoporučují použití pro produkční účely. [45]

4.1 Podporované knihovny a moduly

Vzhledem k tomu, že PyScript je postaven na projektu Pyodide, jsou podporovány veškeré balíčky dostupné v Pyodide (viz Tabulka 1 v příloze P I). Je tomu tak i v případě modulů ze standartní knihovny Pythonu a instalaci balíčků z PyPI.

4.2 Element API

Element API poskytuje způsob, jak jednoduše manipulovat s DOM v HTML dokumentu pomocí PyScriptu. Součástí třídy *Element* je několik vlastností a metod, které umožňují vybrat konkrétní element v HTML a číst nebo upravit jeho obsah.

Vlastnosti třídy *Element*:

- *element* – vrací element s daným id
- *id* – vrací id elementu
- *value* – vrací hodnotu elementu
- *innerHTML* – vrací HTML obsah elementu [46]

Metody třídy *Element*:

- *write* – zapisuje hodnotu *value* elementu
- *clear* – vymaže hodnotu *value* nebo obsah elementu
- *select* – slouží k přístupu k elementu pomocí selektoru
- *add_class* – přidává selektor do atributu *class* elementu
- *remove_class* – odebere selektor z atributu *class* elementu
- *clone* – naklonuje element do nového elementu [46]

4.2.1 Reakce na události

Součástí PyScriptu je velké množství atributů sloužících k obsloužení událostí. Tyto atributy jsou přímo mapovány na události jazyka JavaScript.

Příklady některých atributů:

- *py-click* – kliknutí na element
- *py-onKeyDown* – stisknutí klávesy
- *py-input* – změna v input elementu
- *py-dblclick* – dvojité kliknutí
- *py-mouseover* – přesunutí myši na element [47]

II. PRAKTICKÁ ČÁST

5 PŘÍPRAVA NÁSTROJŮ

5.1 Visual Studio Code

Zkráceně VSCode, je open-source, cross-platform editor zdrojového kódu vyvíjený společností Microsoft. Vývojářům nabízí řadu funkcí pro vytváření a ladění programů, zvýrazňování kódu, IntelliSense (doplňování kódu), ladicí nástroje a integraci se systémem Git. Klíčovou součástí editoru je možnost instalace přídavných rozšíření (extensions). Tato rozšíření pro vizuální úpravy editoru, přidání různých služeb nebo podpory různých programovacích jazyků jsou dostupná na Visual Studio Code Marketplace.

5.1.1 Python

Rozšíření Python umožňuje psát a spouštět Python kód ve VSCode a je základním rozšířením při psaní kódu v tomto jazyce. Přidává spoustu užitečných funkcí jako zvýraznění syntaxe, podporu pro Python IntelliSense, což je funkce, která umožňuje rychlejší a efektivnější psaní kódu tím, že nabízí nápovědu, kontrolu a doplňování syntaxe kódu. Dále rozšíření přidává podporu pro různé verze Pythonu a správu virtuálních prostředí, integrovaný debugger pro Python, podporu pro Jupyter Notebook a další. Dostupné z (https://marketplace.visualstudio.com/items?itemName=ms-python.python).

5.1.2 Live Server

Live Server je rozšíření, které přidává možnost spustit statickou webovou stránku nebo dynamickou aplikaci na lokálním serveru a automaticky se aktualizovat v reálném čase, jakmile se provedou změny v kódu. To znamená, že uživatel nemusí ručně obnovovat stránku, aby viděl změny. Rozšíření dostupné z (https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer).

5.1.3 Pyscript

PyScript rozšíření přidává zvýraznění syntaxe Pythonu vloženého do HTML souboru a šablonu pro vložení Python skriptu. Jedná se pouze o vizuální doplněk pro zjednodušení práce s frameworkem PyScript. Dostupný na adrese (https://marketplace.visualstudio.com/items?itemName=HardeepSingh.pyscript).

5.2 Python

Interpret jazyka Python a jeho rozsáhlá standartní knihovna je k dispozici ke stažení na adrese (https://www.python.org/downloads/). Další možnost může být využití některé z dostupných Python distribucí (např. Anaconda). Seznam dostupných distribucí je uveden zde (https://wiki.python.org/moin/PythonDistributions). Pro spuštění Python kódu v prohlížeči není nutné mít nainstalovaný Python lokálně, avšak je vhodný pro spouštění a otestování napsaných skriptů.

5.3 HTML soubor

Pro to, abychom mohli vytvářet webové aplikace, potřebujeme samozřejmě HTML soubor. Pro uvedené ukázky je použita šablona HTML, kterou je možné v editoru VSCode vygenerovat pomocí zabudovaného rozšíření. Šablona je uvedena níže.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

6 ZÁKLADY PYSCRIPTU

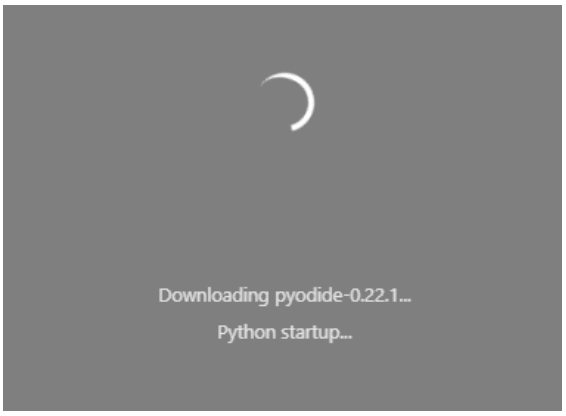
6.1 Integrace do HTML souboru

Framework PyScript není nutné nijak instalovat. Nejjednodušší cesta je „nalinkovat“ potřebné soubory pomocí CDN. Odkazy pro přidání jsou dostupné na oficiálním webu frameworku. Pro připojení k HTML souboru stačí vložit následující řádky do <*head*> tagu:

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://pyscript.net/latest/pyscript.css" />
    <script defer src="https://pyscript.net/latest/pyscript.js"> </script>
    <title>Document</title>
</head>
```

Tím získáme přístup k souborům *pyscript.css* a *pyscript.js*. Soubor CSS je sice nepovinný, avšak poskytuje základní styly pro jednotlivé PyScript komponenty. Soubor *pyscript.js* obsahuje kód v JavaScriptu, který zavádí prostředí Pythonu do prohlížeče a přidává speciální komponenty a elementy.

To, že je vše správně nastavené lze poznat podle načítací obrazovky (splashscreen). Splashscreen je jeden z pluginů PyScriptu, který je ve výchozím nastavení aktivní. Jeho nastavení je možné měnit a zůstává aktivní po dobu načítání potřebných zdrojů PyScriptu.



Obrázek 14 – PyScript splashscreen

6.2 Element <py-script>

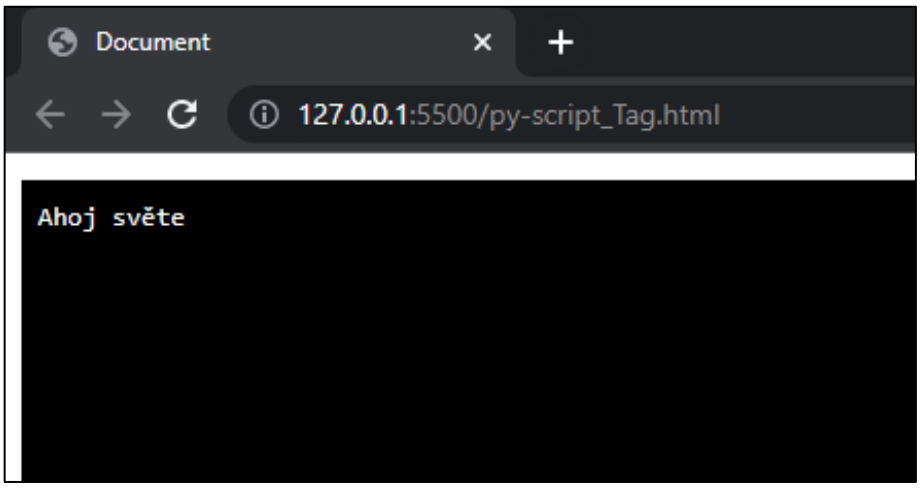
Jedná se o zcela klíčový element, který umožnuje zapsat Python kód přímo do HTML. Existují dvě možnosti, jak toho docílit.

6.2.1 Kód uvnitř <py-script> elementu

Samotný Python kód, je možné psát přímo uvnitř *<py-script>* elementu, který je vložen uvnitř *<body>*. Pro ukázku si uvedeme jednoduchý výpis textu pomocí funkce *print()*.

```
<body>
    <py-script>
        print("Ahoj světe")
    </py-script>
</body>
```

Po zobrazení stránky vypadá výstup následovně:

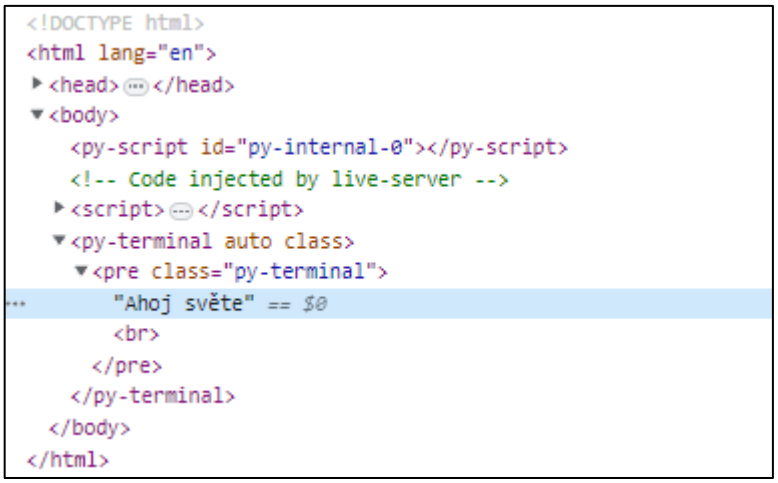


Obrázek 15 – Výpis textu pomocí funkce *print()*

Výstup je velice podobný běžnému výpisu do konzole. To je způsobeno tím, že při použití funkce *print()* je do HTML přidán nový element *<py-terminal>*, a text je umístěn uvnitř tohoto elementu (viz Obrázek 17). Jak už název napovídá, tento element simuluje klasickou konzoli. Nelze do ní však nic psát, slouží tedy jen pro výpis standartního výstupu (stdout) a standartního chybového výstupu (stderr) v Pythonu. Jedná se opět o jeden z pluginů PyScriptu, který je v základním nastavení aktivní a jeho chování je možné měnit. Konkrétní vzhled terminálu je definován v *pyscript.css* souboru, který je uveden v hlavičce HTML.

```
.py-terminal {
    min-height: 10em;
    background-color: black;
    color: white;
    padding: 0.5rem;
    overflow: auto;
}
```

Obrázek 16 – CSS class selektor *.py-terminal*

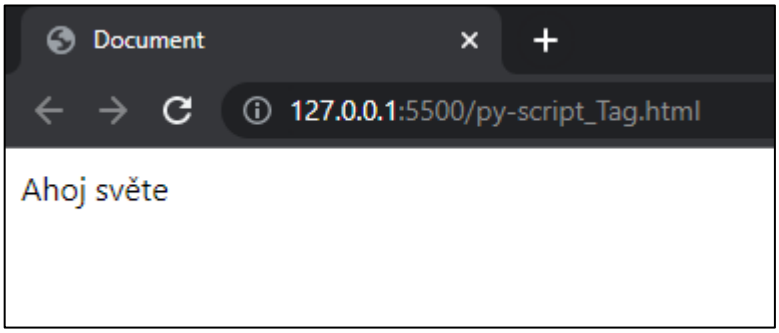


Obrázek 17 – HTML po spuštění skriptu s funkcí *print()*

Alternativní funkce, která umožňuje zobrazit výstup, je funkce *display()*. V tomto případě je text zobrazen normálně a vložen mezi <*div*> element.

```
<body>
    <py-script>
        display("Ahoj světe")
    </py-script>
</body>
```

Výpis textu funkcí *display()* je zobrazen níže:



Obrázek 18 – Výpis textu pomocí funkce *display()*

6.2.2 Kód v externím Python souboru

Pro přidání Python kódu z externího souboru je potřeba uvést v elementu <*py-script*> atribut *src*, kde uvedeme název konkrétního Python souboru.

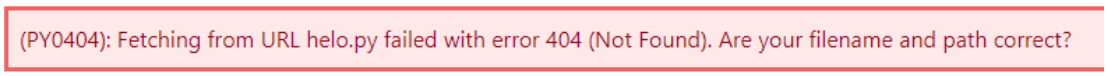
```
<body>
    <py-script src="hello.py"></py-script>
</body>
```

Alternativně lze zkrátit zápis následovně:

```
<body>
    <py-script src="hello.py"/>
</body>
```

Volaný Python soubor *hello.py* obsahuje pouze funkci *display("Ahoj světe")*. Výsledek je tedy stejný jako v přechozím příkladu (viz Obrázek 18).

V případě neexistujícího, či nesprávně zapsaného názvu externího souboru je po spuštění vypsána chybová hláška (viz Obrázek 19).



Obrázek 19 – Chybová hláška při nenalezení souboru

6.3 Python cyklus v HTML souboru

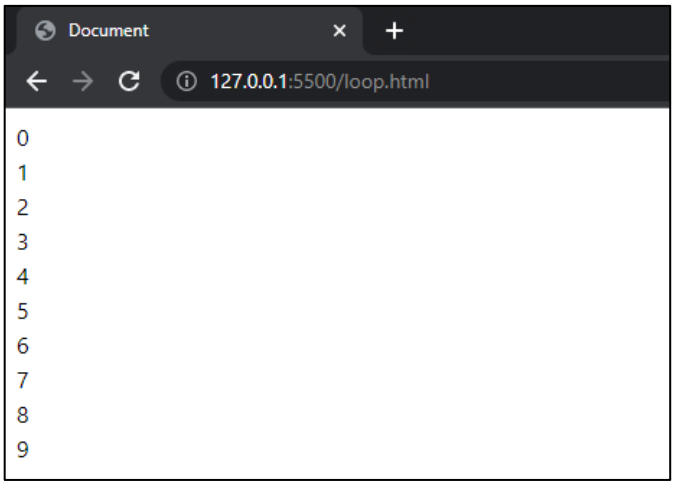
V následující ukázce kódu je do HTML souboru implementován jednoduchý cyklus *for* v Pythonu, který vypíše čísla od 0 do 9. Samotný kód, jak již bylo zmíněno, je možné mít v externím samostatném modulu, na který se odkazuje pomocí atributu *src*. V tomto případě je kód přímo součástí HTML dokumentu.

```
<body>
    <py-script>
        for i in range(10):
            display(i)
    </py-script>
</body>
```

Pro výpis je použita již zmíněná funkce *display()*. Každé číslo je tak součástí svého <*div*> elementu a čísla jsou zobrazena pod sebou (viz Obrázek 20). Při použití klasické funkce *print()* jsou čísla vypsána do „konzole“ v <*py-terminal*> elementu.

Podobně je možné použít cyklus *while*, jehož výstup bude naprosto totožný.

```
<py-script>
    i = 0
    while i < 10:
        display(i)
        i+=1
</py-script>
```



Obrázek 20 – Výpis čísel pomocí for cyklu v Pythonu

6.3.1 Odsazení kódu v <py-script> elementu

Stejně tak jako v běžném Python souboru, je i v případě psaní kódu uvnitř <*py-script*> elementu nutné dodržovat základní pravidla syntaxe. A to především odsazení. Ačkoliv lze samotný element <*py-script*></*py-script*> umístit téměř jakkoliv, veškerý obsah uvnitř těchto tagů, tedy samotný Python kód, je pro správné fungování nutné psát dle pravidel. S odsazením celého kódu si však PyScript poradí a umožní tento kód spustit.

Na obrázku níže je příklad zápisu kódu se správným odsazením ve *for* cyklu, avšak celý kód je odsazen. V běžném Python souboru by s takovým odsazením nešel kód spustit. V tomto případě PyScript kód bez problému spustí. Výstup je stejný jako v předešlém příkladě (viz Obrázek 20).

```
<body>
        <py-script>
        for i in range(11):
            print(i)
                    </py-script>
</body>
</html>
```

Obrázek 21 – Odsazení kódu v <py-script> elementu

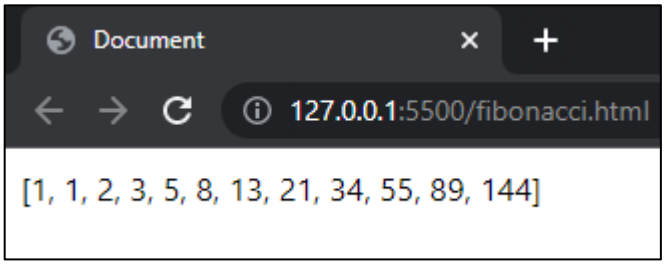
6.4 Výpis výsledku funkce do vlastního elementu

Pro následující příklad použijeme funkci pro zobrazení prvních *n* členů Fibonacciho posloupnosti. Parametr *n* udává počet získaných členů posloupnosti, které jsou vráceny v datovém typu *list*.

Funkce vypadá následovně:

```
def fibonacci(n):
        sequence = []
        a, b = 0, 1
        for i in range(n):
            sequence.append(b)
            a, b = b, a + b
        return sequence
```

Při běžném vypsání pomocí funkce *display()* se zobrazí posloupnost ve formě listu:



Obrázek 22 – Výpis posloupnosti funkcí *display()*

PyScript však umožňuje směrovat výpis do libovolného HTML elementu. K tomu je možné využít volitelný parametr *target* funkce *display()*.

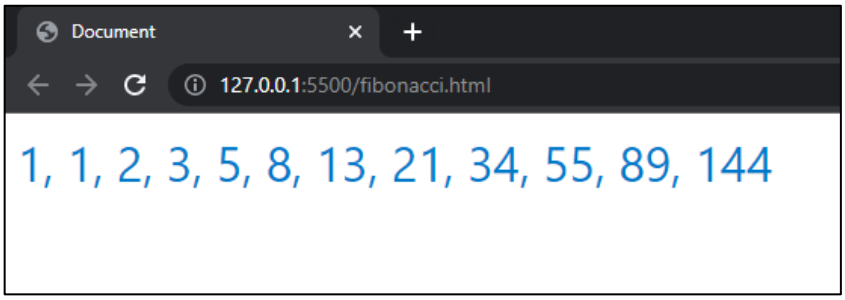
```
<body>
<div id="result" style="color:#007acc; font-size:2em">
</div>
<py-script>
    def fibonacci(n):
        sequence = []
        a, b = 0, 1
        for i in range(n):
            sequence.append(b)
            a, b = b, a + b
        return sequence

    numbers = fibonacci(12)
    strings = map(str, numbers)
    strings = ", ".join(strings)
    display(strings, target="result")
</py-script>
```

Tento kód definuje HTML stránku s jedním <*div*> elementem a identifikátorem *result*, který má nastavenou barvu a velikost písma pro lepší čitelnost. Poté se používá PyScript kód, který vypočítá prvních 12 členů Fibonacciho posloupnosti pomocí funkce *fibonacci()* a uloží je do

seznamu *numbers*. Každé číslo v seznamu je převedeno na řetězec pomocí funkce *str()* a výsledné řetězce jsou uloženy zpět do seznamu *strings* pomocí funkce *map()*.

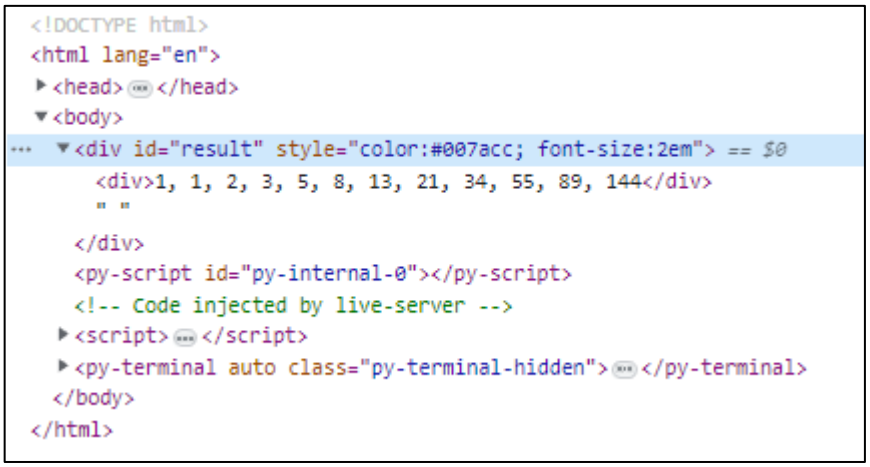
Nakonec se výsledek uloží do HTML elementu s identifikátorem *result* pomocí funkce *display()* s argumenty *strings* (obsahující vypočtené řetězce) a *target="result"* (specifikující identifikátor HTML elementu, do kterého se má výsledek uložit).

Obdobným způsobem lze použít i tzv. *Element API*, kdy místo funkce *display()* je používán následující zápis: *Element('result').write(strings)*. Metoda *write()* objektu *Element('result')*, zapíše výsledek do požadovaného HTML elementu. Výsledek kódu je zobrazen na následujícím obrázku.



Obrázek 23 – Výpis do vlastního <*div*> elementu

O tom, že je výpis součástí našeho elementu se můžeme přesvědčit kromě vzhledu výpisu i ve struktuře spuštěného HTML souboru:



Obrázek 24 – Struktura HTML souboru u Fibonacciho sekvence

Samotná funkce *fibonacci()* může být i součástí externího modulu. V tomto případě je nutné funkci z modulu importovat. To, jakým způsobem importovat a používat moduly, ať už se jedná o moduly standartní knihovny Python, vlastní vytvořené moduly, nebo moduly třetích stran (Numpy, Matplotlib atd.), je vysvětleno v následující kapitole.

7 POUŽÍVÁNÍ MODULŮ A KNIHOVEN

7.1 Element <py-config>

Tento element nám umožňuje základní nastavení závislostí, tedy přístupu k externím modulům a nastavení dalších obecných metadat. Element se umísťuje uvnitř <*body*> elementu a je zapisován ve formátu TOML, popřípadě JSON. Typ konkrétního formátu lze nastavit pomocí atributu *type*, přičemž při jeho neuvedení je základním formátem TOML. Zároveň je možné na daný konfigurační soubor odkazovat atributem *src*.

Na příkladu níže je uvedena konfigurace pluginu <*py-splashscreen*>. Konkrétně jde o zrušení zobrazení načítací obrazovky PyScriptu při spuštění HTML souboru.

```
<py-config type="json" src="">
{
    "splashscreen":{
        "enabled": false
    }
}
</py-config>
```

7.2 Vlastní moduly

Abychom mohli v kódu přistupovat k vlastním modulům, musíme do elementu <*py-config*> definovat k těmto modulům cestu.

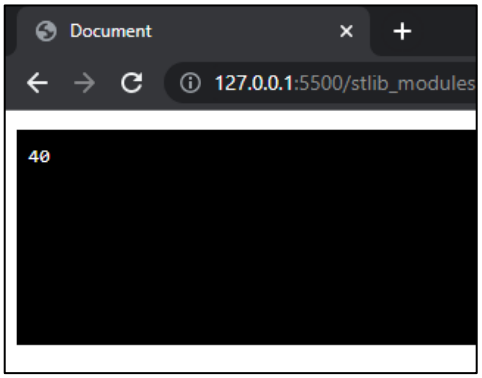
```
<body>
    <div id="result" style="color:#007acc; font-size:2em">
    </div>

    <py-config>
        [[fetch]]
        files=["./fibonacci.py"]
    </py-config>

    <py-script>
        from fibonacci import *

        numbers = fibonacci(12)
        strings = map(str, numbers)
        strings = ", ".join(strings)
        Element('result').write(strings)
</py-script>
</body>
```

Na předchozím příkladu je ukázka již známé funkce pro výpočet Fibonacciho sekvence. V tomto případě je však funkce součástí externího Python modulu. Pro přístup k samotné funkci je potřeba tento modul naimportovat do našeho kódu: *from fibonacci import* *. Modul ale musíme předem načíst v <*py-config*> elementu. V konfiguraci je použito *[[fetch]]*, což umožňuje načíst jeden nebo více souborů, jejichž cesta, vzhledem k HTML souboru, je uvedena v seznamu v položce *files*.

7.3 Moduly ze standartní knihovny Python

U většiny modulů, které jsou součástí standartní knihovny a jsou poskytovány s pomocí projektu Pyodide, není potřeba moduly předem načíst. Tyto moduly stačí jednoduše naimportovat pomocí klíčového slova *import* v rámci kódu v Pythonu.

```
<body>
    <py-script>
        import random
        print(random.randint(1,100))
    </py-script>
</body>
```



Obrázek 25 – Použití modulu random

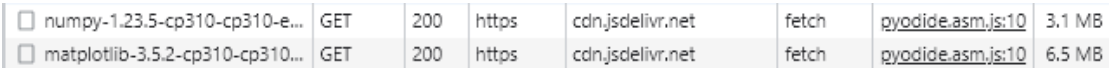
7.4 Moduly třetích stran

Jak již bylo zmíněno, PyScript umožňuje díky projektu Pyodide přistupovat k některým velmi populárním knihovnám. K použití těchto knihoven v kódu je opět nutné nastavit závislosti v elementu <*py-config*>. Následně jen danou knihovnu importujeme přímo v kódu.

Příklad níže demonstruje nastavení závislostí pro knihovny NumPy a Matplotlib a import těchto knihoven přímo do kódu. Obdobným způsobem můžeme přidat další knihovny, které jsou v PyScriptu podporovány (viz Tabulka 1 v příloze P I).

```
<body>
    <py-config>
        packages =["numpy", "matplotlib"]
    </py-config>
    <py-script>
        import numpy as np
        import matplotlib.pyplot as plt
    </py-script>
</body>
```

Po spuštění HTML souboru a poměrně delším nahrávání prostředí dojde k načtení všech potřebných souborů, které umožňují knihovny použít.



Obrázek 26 – Načtení knihovny NumPy a Matplotlib

7.5 Práce s maticemi v knihovně NumPy

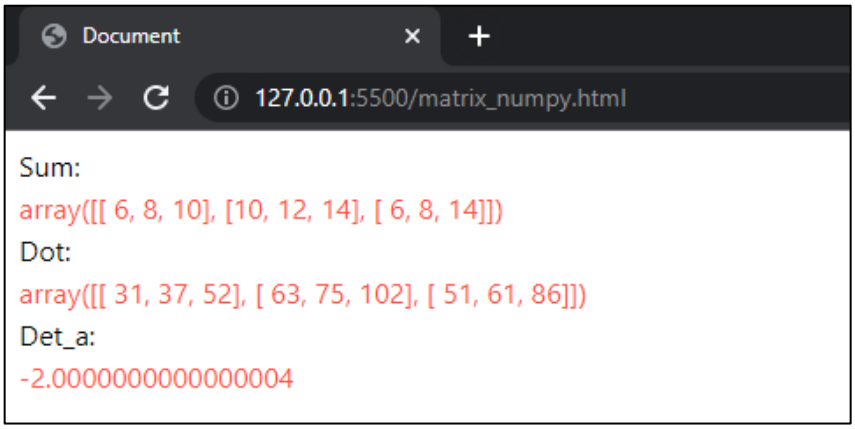
V následující ukázce je příklad použití knihovny NumPy v praxi. Nejprve knihovnu definujeme v elementu <*py-config*>, aby došlo k načtení všech potřebných souborů.

```
<py-config>
    packages =["numpy"]
</py-config>
```

Poté následuje blok Python kódu ohraničený tagy <*py-script*>. Tento kód importuje knihovnu NumPy a vytváří dvě matice, "*a*" a "*b*", které jsou použity pro výpočty součtu, násobení a také determinantu.

```
<py-script>
    import numpy as np

    a = np.array([[1, 2, 3], [3, 4, 5], [2, 3, 5]])
    b = np.array([[5, 6, 7], [7, 8, 9], [4, 5, 9]])

    add = a+b
    Element('result_sum').write(sum)

    dot = np.dot(a,b)
    Element('result_dot').write(dot)

    det_a = np.linalg.det(a)
    Element('result_det_a').write(det_a)
</py-script>
```

Konkrétně jsou matice "*a*" a "*b*" vytvořeny pomocí funkce *np.array()*, která vytvoří pole s danými hodnotami. Po provedení výpočtů jsou výsledky uloženy v proměnných *add*, *dot* a *det_a*. Tyto proměnné jsou následně použity k zápisu výsledků na HTML stránku, pomocí již zmíněné metody *write()*, do konkrétních elementů.



Obrázek 27 – Výpočty s maticemi

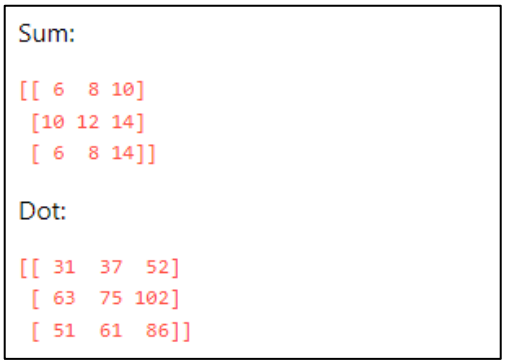
Výsledné matice nejsou zrovna v čitelném formátu. Pro převod matice do čitelnější podoby lze použít funkci *array2string()*, která je součástí knihovny NumPy. Tato funkce převede matici na textový řetězec.

```
Element('result_sum').write(np.array2string(add))
```

Následně je potřeba, jako cílový element pro výpis, zvolit tag <*pre*>. Tento element definuje formátovaný text a zachová veškeré mezery i zalomení řádků.

```
<div>Sum:</div>
<pre id="result_sum" style="color:#ff4f42"></pre>
<div>Dot:</div>
<pre id="result_dot" style="color:#ff4f42"></pre>
```

Po těchto úpravách vypadají matice následovně:



Obrázek 28 – Zobrazení matic v čitelné podobě

Kompletní zdrojový kód příkladu je k dispozici na obrázku níže:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://pyscript.net/latest/pyscript.css" />
    <script defer src="https://pyscript.net/latest/pyscript.js"></script>
    <title>Document</title>
</head>
<body>
    <py-config>
        packages =["numpy"]
    </py-config>
    <div>Sum:</div>
    <pre id="result_sum" style="color:#ff4f42"></pre>
    <div>Dot:</div>
    <pre id="result_dot" style="color:#ff4f42"></pre>
    <div>Det_a:</div>
    <div id="result_det_a" style="color:#ff4f42"></div>
    <py-script>
        import numpy as np

        a = np.array([[1, 2, 3], [3, 4, 5], [2, 3, 5]])
        b = np.array([[5, 6, 7], [7, 8, 9], [4, 5, 9]])

        add = a+b
        Element('result_sum').write(np.array2string(add))

        dot = np.dot(a,b)
        Element('result_dot').write(np.array2string(dot))

        det_a = np.linalg.det(a)
        Element('result_det_a').write(det_a)
    </py-script>
</body>
</html>
```

Obrázek 29 – Práce s maticemi v knihovně NumPy

7.6 Zobrazení grafů pomocí knihovny Matplotlib

Další velmi významnou knihovnou, která je podporována v PyScriptu, je knihovna Matplotlib. Tato knihovna slouží především k vizualizaci dat a tvorbě grafů.

Přístup ke knihovně Matplotlib získáme opět jejím přidáním do seznamu *packages* v <*py-config*> elementu.

```
<py-config>
    packages =["numpy", "matplotlib"]
</py-config>
```

Kód uvedený níže vytváří graf funkce sinus. Nejprve se importují potřebné knihovny. Konkrétně *matplotlib.pyplot* a *numpy*. Poté se pomocí funkce *np.linspace()* vytvoří pole hodnot *x* od 0 do 4π s krokem 0.01. Toto pole bude sloužit jako hodnoty na x-ové ose grafu.

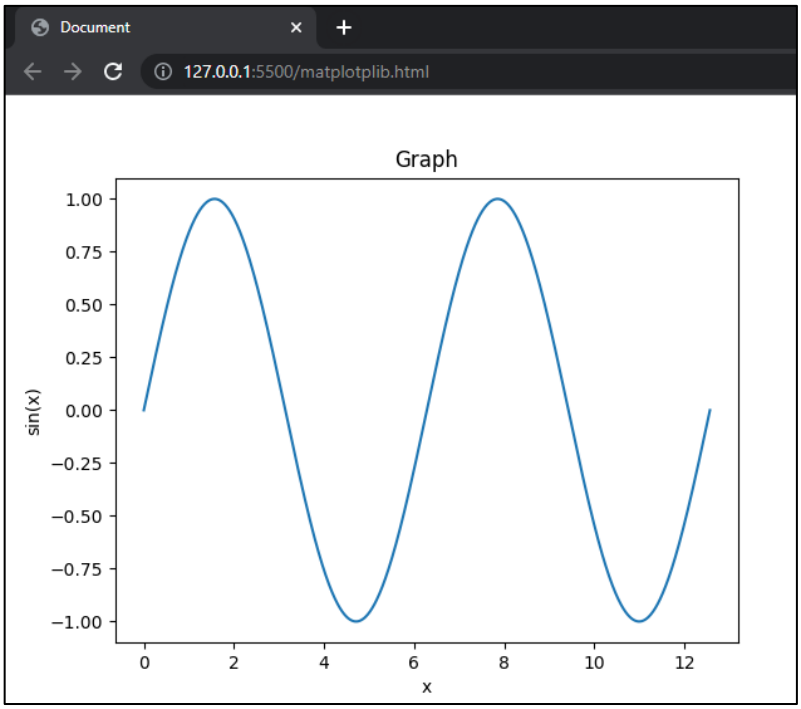
Další řádek vytvoří pole hodnot *y* pomocí funkce *np.sin()*. Toto pole bude sloužit jako hodnoty na y-ové ose grafu.

Pomocí funkce *plt.plot()* se vytvoří graf s hodnotami *x* a *y*. Funkce *plt.xlabel()*, *plt.ylabel()* a *plt.title()* slouží k nastavení popisků os a titulku grafu.

Nakonec se graf zobrazí pomocí funkce *display()* přímo do HTML souboru v prohlížeči.

```
<py-script>
    import numpy as np
    import matplotlib.pyplot as plt
    x = np.linspace(0, 4*np.pi, 1000)
    y = np.sin(x)
    plt.plot(x, y)
    plt.xlabel('x')
    plt.ylabel('sin(x)')
    plt.title('Graph')
    display(plt)
</py-script>
```

Výsledný graf zobrazený v prohlížeči vypadá takto:



Obrázek 30 – Vykreslení grafu přímo do HTML souboru v prohlížeči

7.6.1 Vykreslení 3D grafu

V následující ukázce je kód grafu umístěn v samostatném Python souboru. Z HTML souboru k němu tedy přistoupíme pomocí atributu *src* v elementu <*py-script*>.

```
<py-script src="matplotlib_3D_graph.py">
</py-script>
```

Nejprve jsou načteny všechny potřebné knihovny:

```
import matplotlib.pyplot as plt
import numpy as np
```

Dále se vytvoří data pro graf:

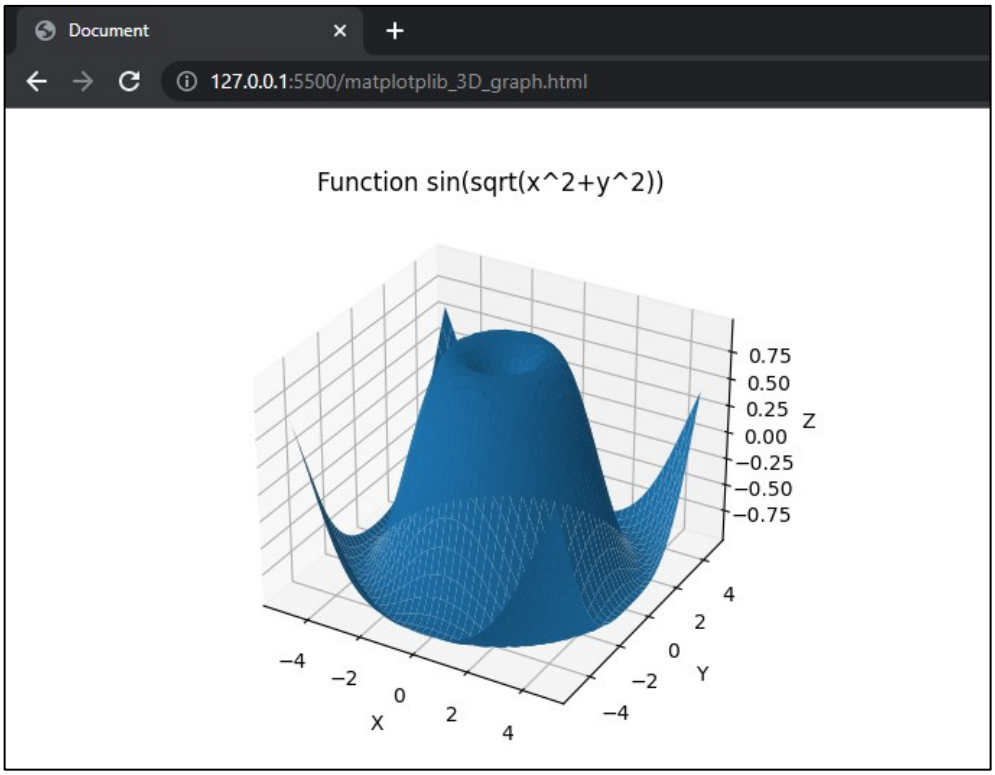
```
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
```

Vytvoří se pole hodnot *X* a *Y*, které obsahují rovnoměrně rozložené hodnoty v rozmezí [- 5,5) s krokem 0.25. Následně se pomocí *np.meshgrid* vytvoří dvě matice *X* a *Y*, které obsahují všechny možné kombinace těchto hodnot (vytvoří se tedy mřížka hodnot X a Y). Podle těchto matic se vypočítá matice *R*, která obsahuje vzdálenost bodů v rovině od počátku souřadnic. Pomocí *np.sin* se nakonec vypočítá matice *Z*, která obsahuje hodnoty funkce *sin(sqrt(x^2+y^2))*.

Nyní je možné vytvořit samotný graf, nastavit název a popis jednotlivých os:

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Function sin(sqrt(x^2+y^2))')
```

Výsledný graf je v HTML souboru vykreslen pomocí funkce *display()*.



Obrázek 31 – Vykreslení 3D grafu přímo do HTML souboru v prohlížeči

matplotlib_3D_graph.py > ...

```
1   import matplotlib.pyplot as plt
2   import numpy as np
3   # data
4   X = np.arange(-5, 5, 0.25)
5   Y = np.arange(-5, 5, 0.25)
6   X, Y = np.meshgrid(X, Y)
7   R = np.sqrt(X**2 + Y**2)
8   Z = np.sin(R)
9   # graph
10  fig = plt.figure()
11  ax = fig.add_subplot(111, projection='3d')
12  ax.plot_surface(X, Y, Z)
13
14  # graph labels
15  ax.set_xlabel('X')
16  ax.set_ylabel('Y')
17  ax.set_zlabel('Z')
18  ax.set_title('Function sin(sqrt(x^2+y^2))')
19
20  display(plt)
```

Obrázek 32 – Zdrojový kód 3D grafu

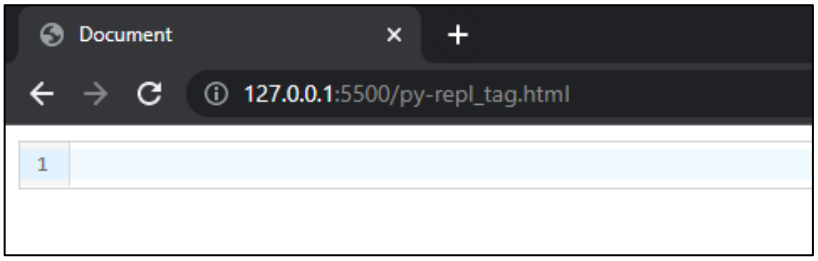
8 READ-EVAL-PRINT-LOOP

Read-eval-print-loop neboli REPL je interaktivní příkazový řádek umožňující zadávat příkazy, popřípadě i větší bloky kódu, které jsou ihned vykonávány a jejich výstup si je možné zobrazit. K těmto účelům je v PyScriptu k dispozici <*py-repl*> element, který přidává tuto interaktivní příkazovou řádku do HTML souboru.

8.1 Element <py-repl>

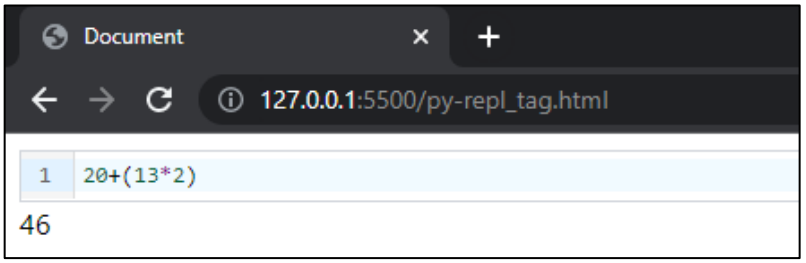
Tuto párovou značku jednoduše vložíme do <*body*> elementu v HTML souboru. Výsledkem je zobrazení vstupu do smyčky REPL, do kterého již můžeme psát samotné výrazy a kód v jazyku Python (viz Obrázek 33).

```
<body>
  <py-repl></py-repl>
</body>
```



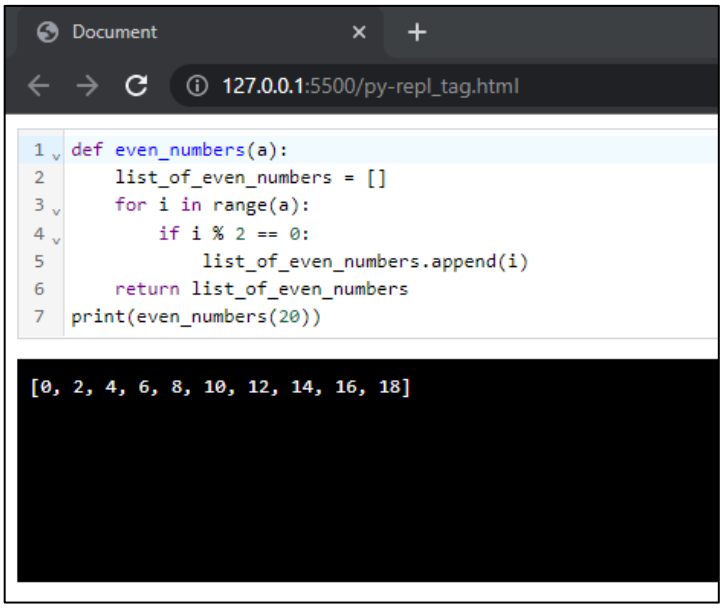
Obrázek 33 – Webová stránka s elementem <*py-repl*>

Provedení zadaného výrazu se vykoná stisknutím klávesové zkratky *Shift+Enter*, popřípadě kliknutím na zelenou šipku umístěnou na konci řádku. Výsledek výrazu je zobrazen ihned pod vstupním řádkem.



Obrázek 34 – Vyhodnocení jednoduchého výrazu pomocí REPL

Kromě jednoduchých výrazů je možné vkládat i víceřádkový kód. Nový řádek se vkládá pomocí klávesy *Enter*.



Obrázek 35 – Vyhodnocení delšího zápisu kódu pomocí REPL

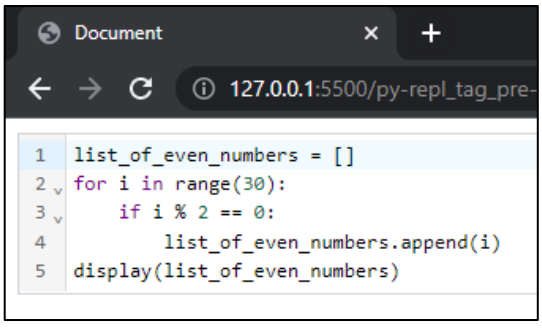
Na obrázku výše je příklad zápisu jednoduché funkce (*even_numbers()*) pro naplnění listu sudými čísly (*list_of_even_numbers*) ve zvoleném rozsahu (parametr funkce *a*).

8.2 Předvyplněný REPL a použití modulů

Mezi samotné tagy <*py-repl*> lze taktéž umístit kód:

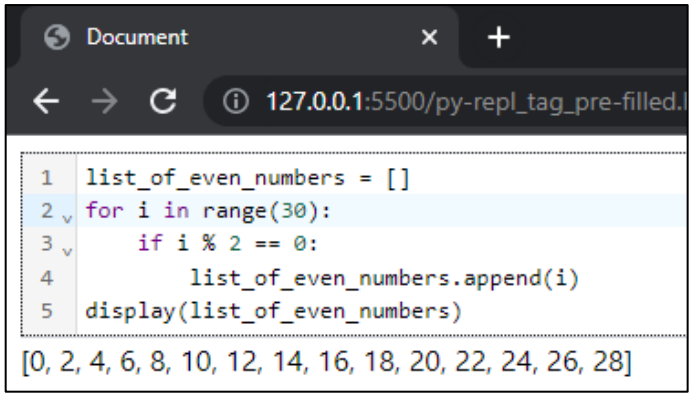
```
<body>
  <py-repl>
        list_of_even_numbers = []
        for i in range(30):
            if i % 2 == 0:
                list_of_even_numbers.append(i)
        display(list_of_even_numbers)
  </py-repl>
</body>
```

Webová stránka v tomto případě zobrazí uvedený kód jako součást smyčky REPL:



Obrázek 36 – Zobrazení předvyplněné smyčky REPL

Tento kód je poté možné samozřejmě spustit:



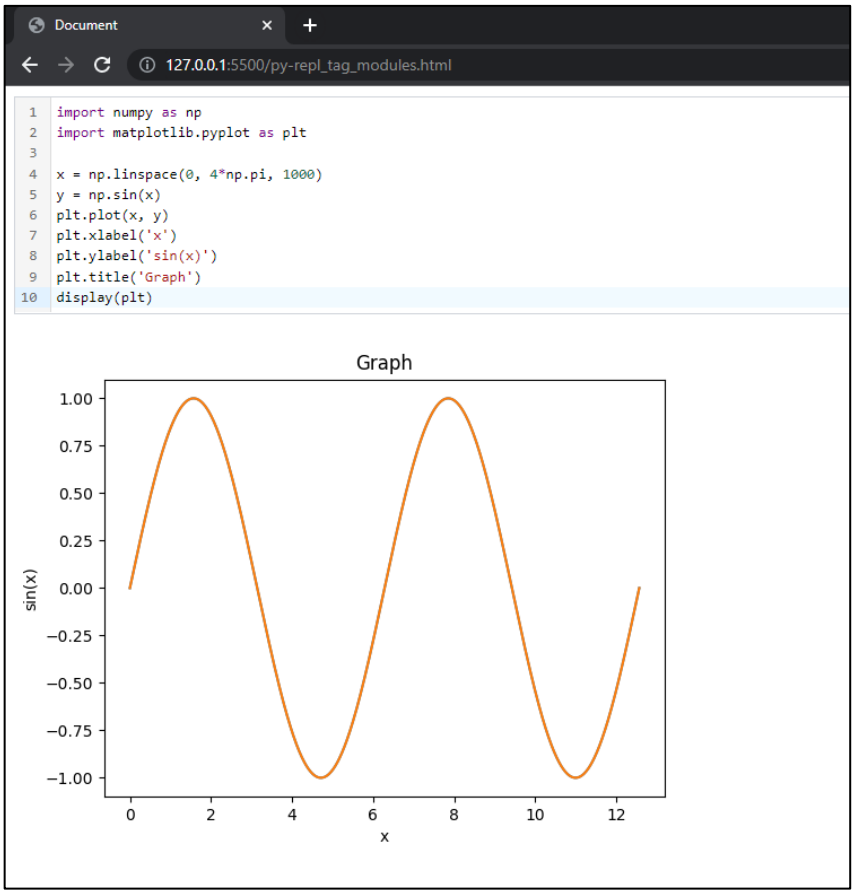
Obrázek 37 – Výsledek spuštění předvyplněné smyčky REPL

8.2.1 Moduly v REPL

Při používání různých modulů v rámci REPL je nutné tyto moduly opět definovat v elementu *<py-config>* a následně importovat v kódu.

```
<body>
    <py-config>
        packages =["numpy", "matplotlib"]
    </py-config>
  <py-repl>
        import numpy as np
        import matplotlib.pyplot as plt

        x = np.linspace(0, 4*np.pi, 1000)
        y = np.sin(x)
        plt.plot(x, y)
        plt.xlabel('x')
        plt.ylabel('sin(x)')
        plt.title('Graph')
        display(plt)
  </py-repl>
</body>
```

Mezi značkami *<py-repl>* je vložen kód pro vykreslení grafu funkce sinus. Po zobrazení HTML souboru bude výsledkem opět předvyplněná REPL smyčka, po jejímž spuštění dojde k vykreslení samotného grafu (viz Obrázek 38).



Obrázek 38 – Vykreslení funkce sinus pomocí REPL

REPL lze také kombinovat společně s elementem <*py-script*>, kdy samotný kód může být schován v rámci tohoto elementu a k jeho volání dojde pomocí REPL.

Funkce pro vykreslení grafu je umístěna mezi značky <*py-script*>. Volání této funkce, konkrétně tedy *display(plot())*, je součástí elementu <*py-repl*>.

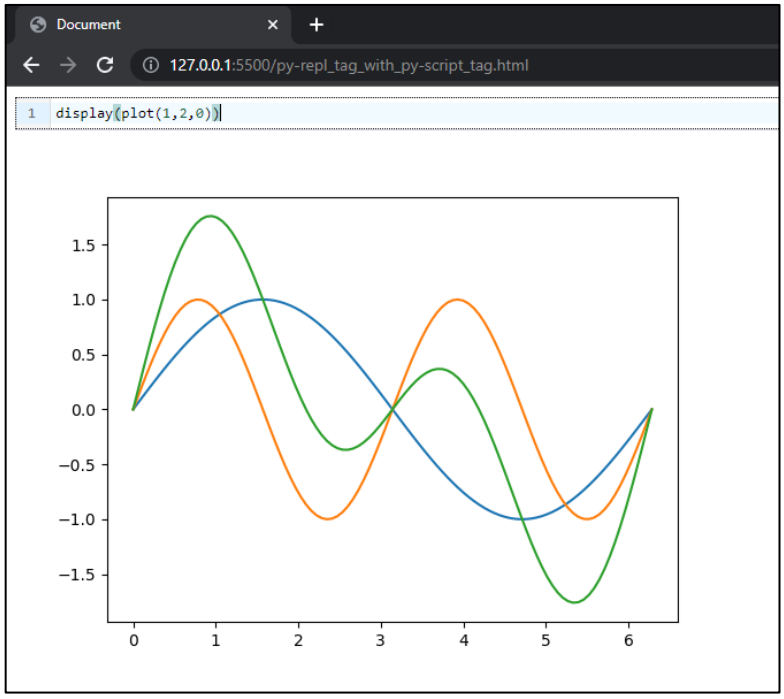
```
<py-script>
    import numpy as np
    import matplotlib.pyplot as plt

    def plot(amplitude, frequency, offset):
        x = np.linspace(0, 2 * np.pi, 100)
        y1 = np.sin(x)
        y2 = amplitude * np.sin(x*frequency) + offset
        plt.clf()
        plt.plot(x, y1, x, y2, x, y1+y2)
        return plt
</py-script>
<py-repl>
    display(plot(1,2,0))
</py-repl>
```

V tomto případě dojde v HTML souboru pouze k zobrazení předvyplněné smyčky REPL:

```
1  display(plot(1,2,0))
```

Obrázek 39 – Výstup v HTML souboru s předvyplněným REPL

Po spuštění se graf zobrazí:



Obrázek 40 – Zobrazení grafu po spuštění kódu z předvyplněného REPL

Touto cestou lze vytvářet poměrně užitečné interaktivní příklady např. k výukovým účelům, kdy uživatel zadá vlastní parametry a po spuštění se výsledky ihned zobrazí.

9 MANIPULACE S DOM V PYSCRIPTU

9.1 Přístup k HTML elementům

V následujícím příkladu je ukázka přístupu k elementům v HTML dokumentu pomocí *id* a manipulace s atributem *class* elementu.

Výchozí kód HTML stránky je následující:

```
<body>
    <py-script>
    </py-script>
    <h1>Navigation menu</h1>
    <ul id="nav_menu">
        <li class="home">Home</li>
        <li class="about">About</li>
        <li class="services">Services</li>
        <li class="contact">Contact</li>
    </ul>
    <div id="pyscript_text"></div>
</body>
```

Součástí výchozího kódu je nečíslovaný seznam, značka *<ul>*, jeho *id* je *nav_menu*. Každá položka seznamu *<li>* má definován atribut *class* (selektor). Posledním elementem je element * *, jehož *id* je *pyscript_text*.

Python kód umístěný v *<py-script>* elementu:

```
<py-script>
    ul_element = Element("nav_menu")
    display(ul_element)
    ul_element.select('.home').add_class('new_class')
    ul_element.select('.about').remove_class('about')
    div_element = Element("pyscript_text")
    div_element.write("Text from PyScript")
</py-script>
```

K přístupu k jednotlivým elementům se používá třída *Element*. Jednoduchým zápisem se tak dostaneme k samotnému seznamu *<ul>*, pomocí jeho identifikátoru *id="nav_menu"*.

```
ul_element = Element("nav_menu")
```

Element je tímto uložen do proměnné *ul_element*.

```
<pyscript.Element object at 0xe35508>
```

Obrázek 41 – Obsah proměnné *ul_element*

K položkám seznamu se přistupuje právě přes proměnnou *ul_element* (obsahující rodičovský element <*ul*> položek seznamu), a to konkrétně díky metodě *select()*, která umožňuje vybrat potomka <*ul*> elementu za pomoci jeho atributu *class*. Po vybraní konkrétní položky seznamu je použita metoda *add_class()*. Tato metoda přidá danému elementu do atributu *class* nový selektor *new_class*, který pro zřejmý výsledek definuje v rámci CSS barvu a velikost písma.

```
ul_element.select('.home').add_class('new_class')
```

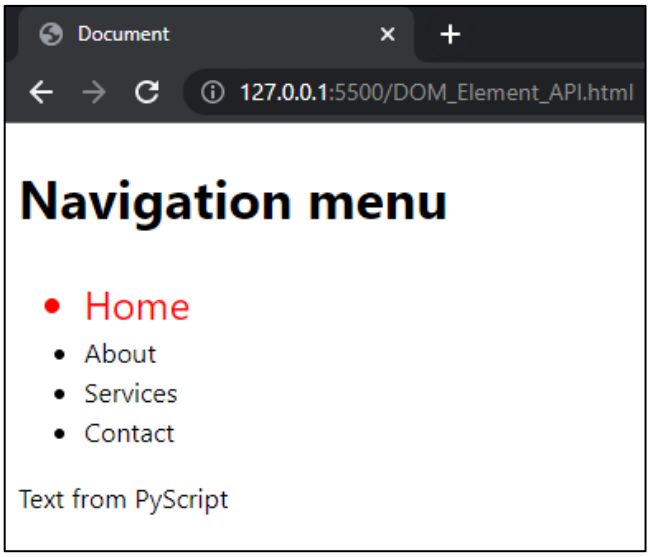
Stejně tak je možné selektor v atributu *class* odebrat:

```
ul_element.select('.about').remove_class('about')
```

Následuje získání <*div*> elementu pomocí *id* a uložení do proměnné *div_element*. Metoda *write()* slouží k modifikaci vlastnosti *innerHTML* (obsahu HTML) elementu. V tomto případě zapíše nový obsah vybraného <*div*> elementu.

```
div_element = Element("pyscript_text")
div_element.write("Text from PyScript")
```

Výsledná stránka je zobrazena níže na obrázku.



Obrázek 42 – Výsledná stránka po modifikaci HTML elementů

Na HTML kódu stránky (viz Obrázek 43) lze vidět, že první položce seznamu byl přidán nový selektor *new_class*. U položky níže byl původní selektor *about* odebrán a element * *, s identifikátorem *pyscript_text*, nyní obsahuje přidaný text pomocí metody *write()*.

Obrázek 43 – HTML kód výsledné stránky po modifikaci HTML elementů

9.2 Zavolání Python funkce po stisku tlačítka

Pro tuto ukázku je použita funkce pro výpočet Fibonacciho posloupnosti uvedené již dříve. Funkce je v tomto případě volána po stisku tlačítka za pomoci atributu *py-click*, jež definuje, co se má stát po kliknutí na daný element.

```
<body>
    <py-script src="calculate_fibonacci_on_click.py">
    </py-script>
     <h3>Click to display Fibonacci sequence:</h3>
    <button py-click="calculate_sequence()">Fibonacci</button>
    <div id="result"></div>
</body>
```

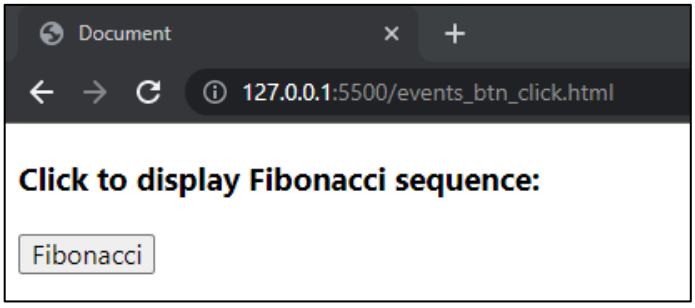
Obsahem HTML souboru je nyní element *<py-script>*, jež odkazuje na soubor v Pythonu. Dále tlačítko *<button>*, které má atribut *py-click="calculate_sequence()"*. Tímto atributem odkazujeme na samotnou funkci *calculate_sequence()*, která je součástí Python skriptu *calculate_fibonacci_on_click.py*. Následuje * * element pro zobrazení výsledků.

Soubor *calculate_fibonacci_on_click.py*:

```
def calculate_sequence():
    def fibonacci(n):
        sequence = []
        a, b = 0, 1
        for i in range(n):
            sequence.append(b)
            a, b = b, a + b
        return sequence
    numbers = fibonacci(12)
    strings = map(str, numbers)
    strings = ", ".join(strings)
    Element('result').write(strings)
```

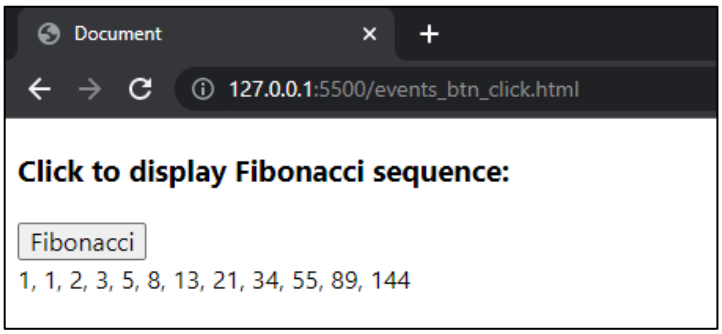
Došlo zde pouze k „obalení“ původní funkce *fibonacci(),* aby ji bylo možné volat po stisku tlačítka.

Zobrazená HTML stránka vypadá takto:



Obrázek 44 – HTML stránka s tlačítkem pro volání Python funkce

Po kliknutí na tlačítko dojde k zavolání funkce *calculate_sequence()*:



Obrázek 45 – HTML stránka po stisknutí tlačítka

9.3 Čtení hodnoty ze vstupu

V předchozím příkladu byl počet vypsaných prvků Fibonacciho sekvence definován přímo v kódu. Nyní tuto hodnotu získáme přímo od uživatele pomocí vstupu <*input*> umístěným v HTML souboru.

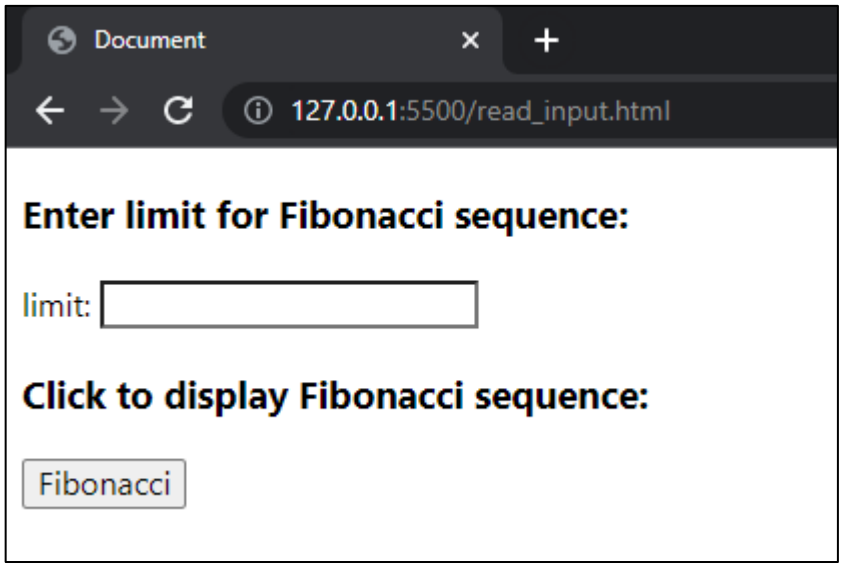
```
<body>
    <py-script src="calculate_fibonacci_on_click_limit.py">
    </py-script>
    <h3>Enter limit for Fibonacci sequence:</h3>
    limit: <input type="number" id='limit'>
    <h3>Click to display Fibonacci sequence:</h3>
    <button py-click="calculate_sequence()">Fibonacci</button>
    <div id="result" ></div>
</body>
```

Hodnotu zapsanou uživatelem v tomto vstupu získáme pomocí třídy *Element*, kde vložíme identifikátor elementu <*input*>, a její vlastnosti *value*. Vstup je nutné převést na číselnou hodnotu funkcí *int()*.

```
limit = int(Element('limit').value)
```

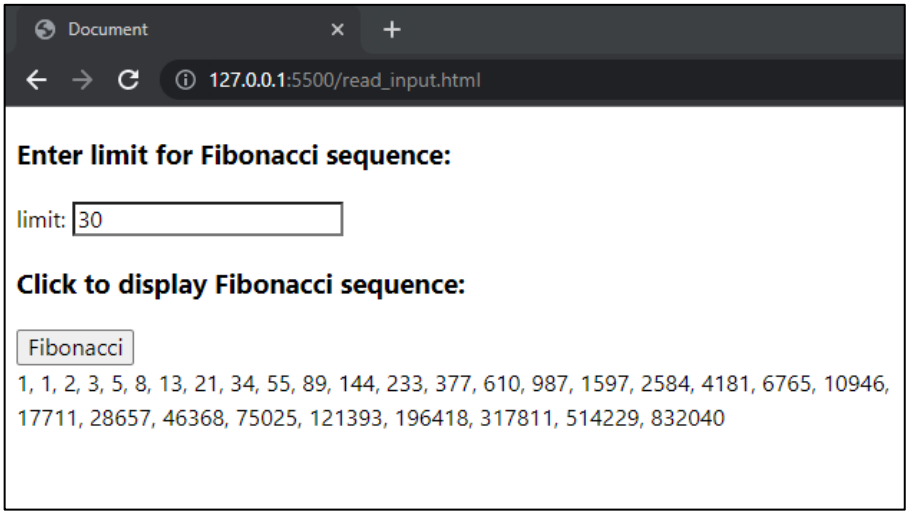
Celá funkce pro obsloužení tlačítka vypadá poté takto:

```
def calculate_sequence():
    limit = int(Element('limit').value)
    def fibonacci(limit):
        sequence = []
        a, b = 0, 1
        for i in range(limit):
            sequence.append(b)
            a, b = b, a + b
        return sequence
    numbers = fibonacci(limit)
    strings = map(str, numbers)
    strings = ", ".join(strings)
    Element('result').write(strings)
```



Obrázek 46 – Zobrazení stránky s inputem

Po zadání číselné hodnoty a stisknutí tlačítka:



Obrázek 47 – Zobrazení stránky po vložení hodnoty limit a stisknutí tlačítka

Volání samotné funkce pro výpis Fibonacciho sekvence jde také aplikovat na samotný element *<input>*. Přidáním atributu *py-input="calculate_sequence()"* elementu *<input>* docílíme, že se funkce zavolá po každé změně hodnoty limitu.

```
<input py-input="calculate_sequence()" type="number" id='limit'>
```

9.4 Interaktivní vykreslení grafu na základě zadaných parametrů

Pro tento příklad je využita předešlá funkce pro vykreslení grafu se zadáním jednotlivých parametrů. Nyní je však na webové stránce umístěn formulář pro zadání těchto parametrů uživatelem.

```
<div class="input-form">
    <div>Amplitude</div>
    <input py-input="plot()" type="number" id="amplitude" value="1" size="5" step="0.1" />
    <div>Frequency</div>
    <input py-input="plot()" type="number" id="frequency" value="1" size="5" step="0.1"/>
    <div>Offset</div>
    <input py-input="plot()" type="number" id="offset" value="0" size="5" step="0.1"/>
    <button py-click="plot()">Show graph</button>
    <div id="graph"></div>
</div>
```

Obrázek 48 – HTML kód formulář pro zadání parametrů

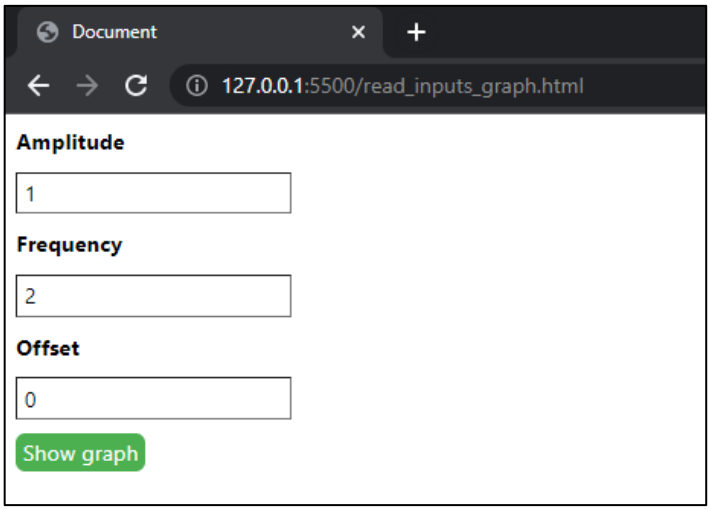
Tento kód představuje jednoduchý formulář s třídou *input-form*, který obsahuje tři vstupní pole pro zadání hodnot amplitudy, frekvence a offsetu, tlačítko pro vykreslení grafu a prázdný <*div*> element pro zobrazení grafu. Jednotlivé vstupy obsahují atribut *py-input*, který zajistí volání funkce pro vykreslení grafu po změně těchto hodnot. Graf se tak v reálném čase bude přepisovat při změně některého z parametrů.

```
<py-script>
    import numpy as np
    import matplotlib.pyplot as plt
    def plot():
        amplitude =float(Element('amplitude').value)
        frequency = float(Element('frequency').value)
        offset = float(Element('offset').value)
        x = np.linspace(0, 2 * np.pi, 100)
        y1 = np.sin(x)
        y2 = amplitude * np.sin(x*frequency) + offset
        plt.clf()
        plt.plot(x, y1, x, y2, x, y1+y2)
        Element('graph').write(plt)
        return plt
</py-script>
```

Obrázek 49 – Funkce *plot()* pro vykreslení grafu ze zadaných parametrů

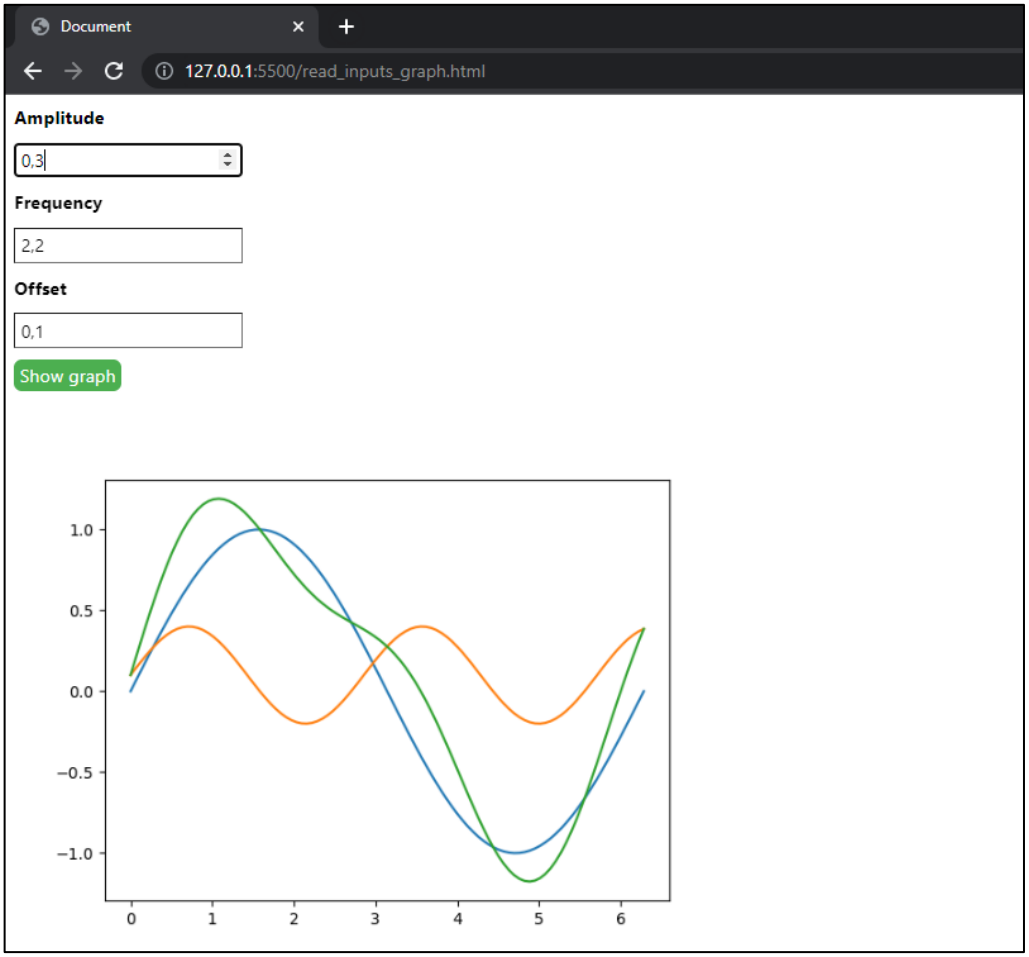
Hodnoty parametrů jsou získány ze vstupních polí třídou *Element* pomocí vlastnosti *value* a převedeny na číselný formát *float*. Takto získané hodnoty jsou použity pro výpočet dat pro samotný graf.

Formulář po menších úpravách stylu zobrazení je uveden na následujícím obrázku.



Obrázek 50 – Zobrazení webové stránky s formulářem vstupních dat pro graf

Po kliknutí na tlačítko, či změně některého z parametrů dojde k zobrazení grafu:



Obrázek 51 – Zobrazení grafu se zadanými parametry

10 PYTHON A JAVASCRIPT

10.1 Předání JavaScript objektů do PyScriptu

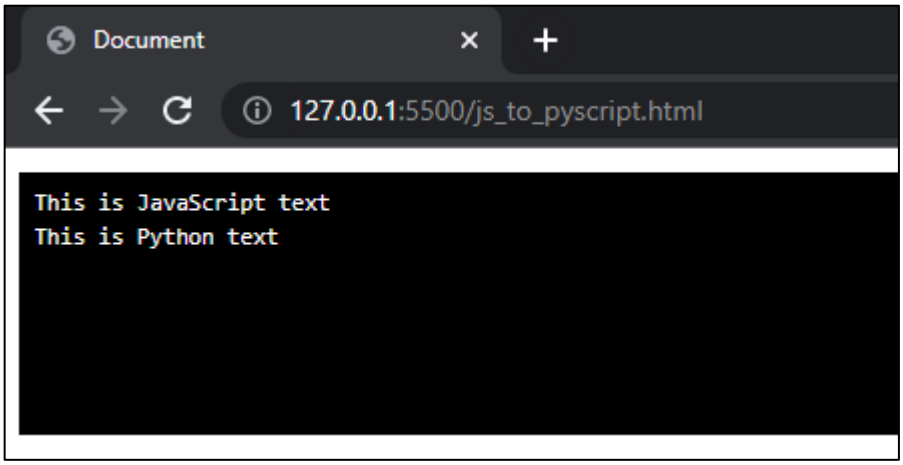
10.1.1 Předání proměnné z JavaScriptu

Na následujícím příkladu je ukázka textové proměnné z jazyka JavaScript, která je předána a vypsána pomocí Pythonu v PyScriptu.

```
<script>
    js_text = "This is JavaScript text"
</script>
```

Kód výše ukazuje vytvoření proměnné *js_text* v JavaScriptu. Následuje již kód psaný v Pythonu. Proměnnou *js_text* je potřeba importovat pomocí zápisu *from js import js_text*. Poté je možné k této proměnné přistupovat a manipulovat v jazyku Python. V kódu níže je hodnota proměnné vypsána funkcí *print()*.

```
<py-script>
    from js import js_text

    python_text = "This is Python text"
    print(js_text)
    print(python_text)
</py-script>
```



Obrázek 52 – Vypsání JavaScript proměnné v PyScriptu

10.1.2 Volání funkce z JavaScriptu

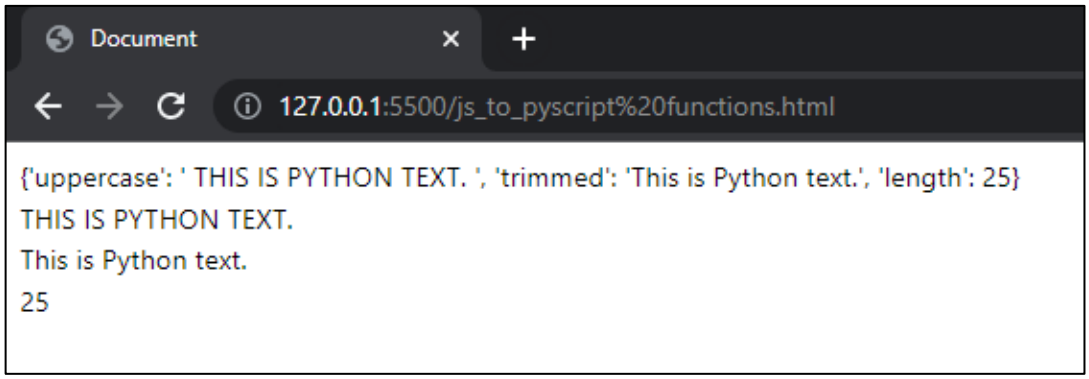
V této ukázce je zobrazen příklad JavaScript funkce. Tato funkce přijímá jako vstupní parametr textový řetězec, se kterým dále manipuluje.

```
<script>
    function manipulateString(input) {
        const uppercase = input.toUpperCase();
        const trimmed = input.trim();
        const length = input.length;
        const result = {
            uppercase: uppercase,
            trimmed: trimmed,
            length: length
        };
        return result;
    }
</script>
```

Funkce *manipulateString()* provádí několik operací s vstupním řetězcem a vrací objekt obsahující výsledky těchto operací. První z nich je převod textového řetězce na velká písmena. Následně jsou metodou *trimm()* odstraněny všechny mezery. Poté je získána délka řetězce vlastností *length*. Je vytvořen JavaScript objekt *result*. Do objektu jsou tyto výsledky uloženy a objekt je funkcí vrácen.

```
<py-script>
    from js import manipulateString
    python_text = "   This is Python text.  "
    result = manipulateString(python_text).to_py()
    display(result)
    display(result['uppercase'])
    display(result['trimmed'])
    display(result['length'])
</py-script>
```

Kód v Pythonu uvedený výše v prvním kroku importuje funkci *manipulateString()* z JavaScriptu do Pythonu pomocí příkazu *from js import manipulateString*. Dále je vytvořena proměnná *python_text*, která obsahuje textový řetězec. Proměnná *result* ukládá výsledek funkce *manipulateString()*. Metoda *to_py()* je použita k převodu objektu z JavaScriptu na ekvivalentní objekt v Pythonu.



Obrázek 53 – Výstup po použití funkce v JavaScriptu

Na obrázku výše lze vidět, že použitá metoda *to_py()* převede objekt, který vrací funkce *manipulateString()*, na Python objekt typu dictionary.

10.2 Předání funkce z Pythonu do JavaScriptu

V následujícím kódu je příklad předání funkce v Pythonu do JavaScriptu. Jedná se o jednoduchou funkci, která vynásobí dvě zadané hodnoty.

Kód funkce v Pythonu vypadá takto:

```
<py-script>
    def multiply_numbers(x, y):
        return x * y
</py-script>
```

Následuje funkce v JavaScriptu:

```
<script>
    function  show_result() {
    const multiplyFunction = pyscript.interpreter.globals.get('multiply_numbers');
    const result = multiplyFunction(5, 7);
    console.log(`result: ${result}`);
}
</script>
```

Obrázek 54 – Kód v JavaScriptu pro volání funkce z Pythonu

Všechny dostupné globální objekty v Pythonu (proměnné, třídy, funkce atd.) lze získat v JavaScript kódu pomocí zápisu *pyscript.interpreter.globals.get()*, kdy jako parametr je uveden v uvozovkách název. V tomto případě je to jméno Python funkce *multiply_numbers*. Odkaz na tuto funkci je tímto uložen do proměnné *multiplyFunction*, která je společně s parametry volána (*multiplyFunction(5, 7)*) a výsledek je uložen do proměnné *result*, který je nakonec vypsán do konzole.

Funkce *show_results()* je volána po kliknutí na tlačítko:

```
<button onclick="show_result()">Multiply</button>
```

Po stisknutí tlačítka je výsledek zobrazen v konzoli:

```
result: 35
```

Obrázek 55 – Výsledek výpočtu v konzoli

10.3 Uchování dat v local storage prohlížeče

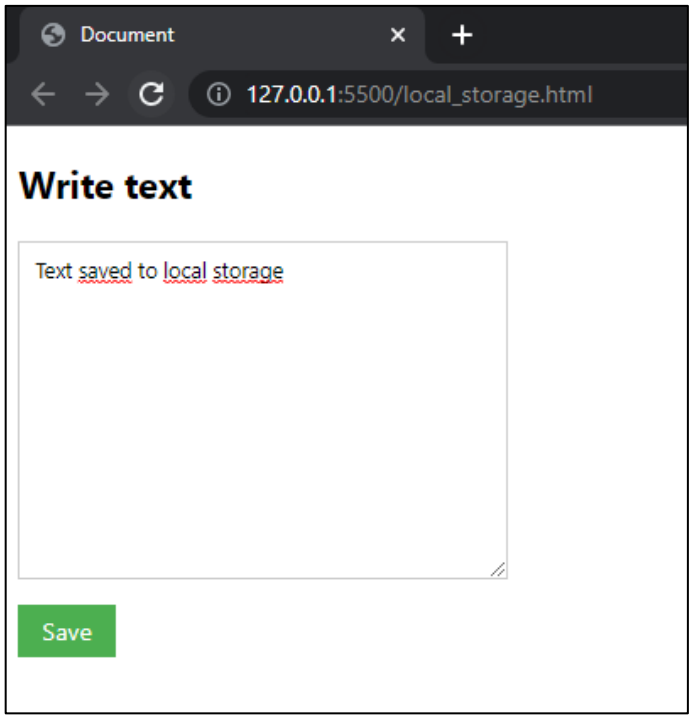
V kódu níže je ukázka uložení obsahu, který je získán z textového pole, do úložiště local storage prohlížeče.

```
33  <body>
34      <h2>Write text</h2>
35      <textarea id="text_area"></textarea>
36      <button id="save" py-click="save_to_localStorage()">Save</button>
37      <py-script>
38          from js import localStorage
39          def save_to_localStorage():
40              text = Element('text_area').value
41              localStorage.setItem("text", text)
42              if localStorage.getItem("text"):
43                  text_input =  Element("text_area")
44                  text_input.write(localStorage.getItem("text"))
45      </py-script>
46  </body>
```

Obrázek 56 – Python kód pro uložení dat do local storage

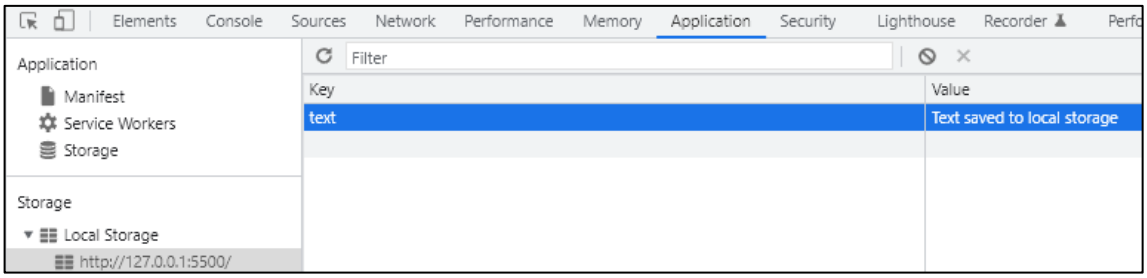
Samotný HTML kód obsahuje pouze element <*textarea*> a tlačítko, které po stisknutí volá funkci *save_to_localStorage()*.

Local storage je importováno z modulu js pomocí *from js import localStorage*. Následuje definice samotné funkce. V ní je do proměnné *text* uložen obsah <*textarea*> elementu. Data v local storage jsou ukládána ve formě *klíč: hodnota*. Toho je docíleno metodou *setItem()*, jejíž první parametr je název klíče a druhý je proměnná *text*, která obsahuje samotná data. Následuje podmínka, která kontroluje, zdali již daný klíč neobsahuje nějaká data. Jestliže ano, tak jsou zapsána metodou *write()* do textového pole. To znamená, že při opětovném otevření webové stránky bude uložený text zobrazen v textovém poli.



Obrázek 57 – Zobrazení stránky s textovým polem ukládající obsah

Po vepsání textu a stisknutí tlačítka, jsou data uložena do local storage prohlížeče.



Obrázek 58 – Data uložená v local storage

10.4 HTTP požadavek v PyScriptu

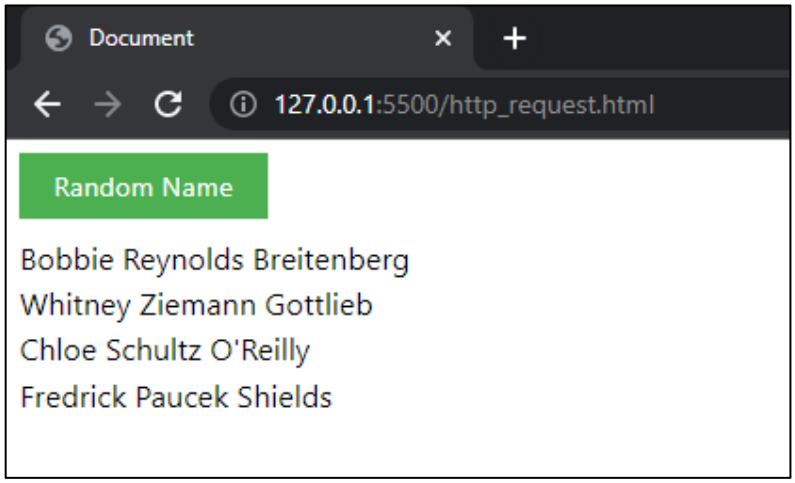
V následujícím příkladu je ukázka použití požadavku GET k získání dat z API. Konkrétně jde o data z adresy https://random-data-api.com/api/name/random_name. Tato API slouží ke generování náhodných dat, v tomto případě náhodných jmen.

K odeslání GET požadavku je použita metoda *pyfetch()*, dostupná v *pyodide.htpp* modulu. Tato metoda umožňuje vytvářet asynchronní požadavky. Dále je potřeba importovat modul *asyncio*, který poskytuje klíčová slova *async* a *wait*.

```
<body>
    <button py-click="get_random_name()">Random Name</button>
    <div id="name"></div>
    <py-script>
        from pyodide.http import pyfetch
        import asyncio
        async def get_random_name():
            url = 'https://random-data-api.com/api/name/random_name'
            response = await pyfetch(url, method="GET")
            data = await response.json()
            output =  f"{data.get('first_name')} {data.get('middle_name')} {data.get('last_name') }"
            Element('name').write(output, append="True")
    </py-script>
</body>
```

Obrázek 59 – Kód zaslání GET požadavku pro získání a zobrazení dat

Následuje vlastní funkce *get_random_name()*, která je volána po stisku tlačítka. Funkce obsahuje proměnnou *url*, která uchovává konkrétní url adresu pro získání dat. Poté je použita metoda *pyfetch()* s parametrem *url* a nastavením požadavku na metodu GET. Získaný objekt odpovědi metodou *pyfetch()* je uložen do proměnné *response*. Metoda *json()* převede tento objekt na objekt v Pythonu a uloží jej do proměnné *data*. Po získání dat jsou jména uložena do proměnných *first_name*, *middle_name* a *last_name*. Tyto proměnné jsou použity k vytvoření výstupního řetězce, který je metodou *write()* zapsán do <*div*> elementu. Nastavením parametru *append* v metodě *write()* zajistíme, aby se nově vygenerovaná jména nepřepisovala, ale byla vypsána pod sebou.



Obrázek 60 – Náhodná jména vygenerována po stisku tlačítka

ZÁVĚR

Bakalářská práce je zaměřena na možnosti využití programovacího jazyka Python pro tvorbu webových aplikací na straně prohlížeče. V úvodu teoretické části se čtenář seznámí se základními vlastnostmi Pythonu, jeho syntaxí a známými implementacemi. Další část je věnována popisu webových aplikací, jejich obecné architektuře, používaným technologiím na straně prohlížeče a roli Pythonu ve webových aplikacích. Hlavní kapitola teoretické části seznámí čtenáře s řešeními, která umožnují použití Pythonu na straně webového prohlížeče. Konkrétně s tzv. transpilací zdrojového kódu nebo použitím formátu WebAssembly, který je schopen běžet ve webovém prohlížeči. Dále jsou popsány konkrétní vybrané projekty, které jsou na těchto řešeních založeny.

Praktická část je věnována frameworku PyScript, který poměrně intuitivným způsobem poskytuje možnost zápisu kódu v Pythonu do HTML souboru. V úvodu praktické části je čtenář seznámen s přípravou prostředí a metodami, jak vložit skript napsaný v Pythonu do HTML souboru. Následují ukázky použití externích modulů a knihoven, jejichž součástí byla i práce s knihovnou NumPy a MatPlotlib pro zobrazení grafů. Poté je čtenář seznámen s interaktivním příkazovým řádkem REPL a možnostmi manipulace s DOM v HTML souboru, které PyScript nabízí. Závěr praktické části popisuje čtenáři interakci mezi jazyky Python a JavaScript společně s ukázkami uložení dat do úložiště prohlížeče a vytvoření HTTP požadavku pro získaní dat z API.

Cílem této práce bylo prozkoumat dostupné projekty, které umožňují vytvářet webové aplikace v Pythonu na straně webového prohlížeče a představit řešení, které tyto projekty používají. Na praktických příkladech poté ukázat možnosti využití frameworku PyScript.

Přínos práce je v přehledu o současných možnostech využití programovacího jazyka Python pro tvorbu webových aplikací a praktických ukázkách použití frameworku PySript.

SEZNAM POUŽITÉ LITERATURY

[1] MASTRODOMENICO, Rob. *The Python Book* [online]. Hoboken: John Wiley & Sons Ltd, 2022 [cit. 2023-04-03]. ISBN 9781119573395. Dostupné z: https://proxy.k.utb.cz/login?url=https://onlinelibrary.wiley.com/doi/book/10.1002/9781119573364

[2] Python: Applications for Python. In: *Python* [online]. Python Software Foundation [cit. 2023-04-03]. Dostupné z: https://www.python.org/about/apps/

[3] OSTROWSKA, Kamila. A Brief History of Python. In: *LearnPython* [online]. Vertabelo SA, ©2016-2023 [cit. 2023-04-04]. Dostupné z: https://learnpython.com/blog/history-of-python/

[4] Status of Python Versions: Python Release Cycle. In: *Python Developer's Guide* [online]. ©2011-2023 [cit. 2023-04-04]. Dostupné z: https://devguide.python.org/versions/#versions

[5] Python Syntax: Python Indentation. In: *W3Schools* [online]. [cit. 2023-04-03]. Dostupné z: https://www.w3schools.com/python/python_syntax.asp

[6] The Python Standard Library. In: *Python 3.11.2 documentation* [online]. Python Software Foundation, ©2001-2023 [cit. 2023-04-04]. Dostupné z: https://docs.python.org/3/library/index.html

[7] *Python Package index* [online]. Python Software Foundation, 2023 [cit. 2023-04-04]. Dostupné z: https://pypi.org/

[8] *Jython* [online]. [cit. 2023-05-13]. Dostupné z: https://www.jython.org/

[9] *IronPython* [online]. .NET Foundation [cit. 2023-05-13]. Dostupné z: https://ironpython.net/

[10] PyPy - Features. In: *PyPy* [online]. The PyPy Team, 2023 [cit. 2023-05-13]. Dostupné z: https://www.pypy.org/features.html

[11] What is a Web Application?. In: *Javapoint* [online]. ©2011-2021 [cit. 2023-04-03]. Dostupné z: https://www.javatpoint.com/web-application

[12] MDN, contributors. Client-Server Overview. In: *MDN web docs* [online]. ©1998–2023 [cit. 2023-04-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview

[13] HTML basics. In: *Developer.mozilla.org* [online]. Mozilla Foundation, ©1998–2023 [cit. 2023-04-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

[14] NIEDERST ROBBINS, Jennifer. *Learning web design: a beginner's guide to HTML, CSS, Javascript, and web graphics*. Fifth edition. Beijing: O'Reilly, 2018. ISBN 978-1-491-96020-2.

[15] ČÁPKA, David. Lekce 1 - Úvod do JavaScriptu: Okolnosti vzniku jazyka. In: *ITnetwork* [online]. Praha: David Čápka, 2023 [cit. 2023-04-07]. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk

[16] PODMOLÍK, Leopold. JAKÝ FRONTEND FRAMEWORK ZVOLIT?: TOP JS frameworky. In: *VisionLabs* [online]. visionslabs, 2023 [cit. 2023-04-10]. Dostupné z: https://visionslabs.io/jaky-frontend-framework-zvolit/

[17] GARG, Manish. Top 11 Python Frameworks for Web Development In 2023. In: *Net Solutions* [online]. Net Solutions, 2023 [cit. 2023-05-15]. Dostupné z: https://www.netsolutions.com/insights/top-10-python-frameworks-for-web-development-in-2019/

[18] Why Use Python for Web Development in 2023? 5 Reasons & Benefits: How Does Python Web Development Work for Real Life Web Development Projects?. In: *Goldsish Code* [online]. Goldfish Code, 2023 [cit. 2023-05-11]. Dostupné z: https://www.goldfishcode.com/blog-posts/python-for-web-development

[19] TIŠNOVSKÝ, Pavel. PyScript: další technologie umožňující využití Pythonu v prohlížeči. In: *Root.cz* [online]. Praha: Internet Info, 2023 [cit. 2023-04-05]. Dostupné z: https://www.root.cz/clanky/pyscript-dalsi-technologie-umoznujici-vyuziti-pythonu-v-prohlizeci/

[20] DIONNE, Mathieu. Reasons Why JavaScript is Omnipresent in Modern Development. In: *Snipcart* [online]. [cit. 2023-04-04]. Dostupné z: https://snipcart.com/blog/why-javascript-benefits#benefits

[21] THORNTON, Scott. What are compilers, translators, interpreters, and assemblers?. In: *Microcontroller tips* [online]. 2023 [cit. 2023-04-04]. Dostupné z: https://www.microcontrollertips.com/compilers-translators-interpreters-assemblers-faq/

[22] TOMASSETTI, Federico. How to write a transpiler. In: *Strumenta* [online]. Torino – Italy [cit. 2023-04-04]. Dostupné z: https://tomassetti.me/how-to-write-a-transpiler/

[23] PRINCE, Dan. Understanding ASTs by Building Your Own Babel Plugin. In: *Sitepoint* [online]. Australia: SitePoint Pty. Ltd., 2023 [cit. 2023-04-05]. Dostupné z: https://www.sitepoint.com/understanding-asts-building-babel-plugin/

[24] O'KANE, Rory. List of languages that compile to JS. In: *GitHub* [online]. GitHub, Inc., 2023 [cit. 2023-04-05]. Dostupné z: https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS#python

[25] BANDARA, Lakshan. What are transpilers and how code transpilation works?. In: *Lakshan-bandara9320.medium* [online]. [cit. 2023-04-05]. Dostupné z: https://lakshan-bandara9320.medium.com/what-are-transpilers-and-how-code-transpiling-works-6cb8270a4b27

[26] Transpiler. In: *Devopedia: for developers, by developers* [online]. [cit. 2023-04-05]. Dostupné z: https://devopedia.org/transpiler

[27] *WebAssembly* [online]. World Wide Web Consortium, 2023 [cit. 2023-04-10]. Dostupné z: https://webassembly.org/

[28] GALLANT, Gerard. *WebAssembly in Action*. Shelter Island (New York): Manning, 2019. ISBN 978-1617295744.

[29] MCCONNELL, Judy. WebAssembly support now shipping in all major browsers. In: *Blog.mozilla.org* [online]. Mozilla Corporation, ©1998–2023 [cit. 2023-04-11]. Dostupné z: https://blog.mozilla.org/en/mozilla/webassembly-in-browsers/

[30] VAN DER HIEL, Amy. World Wide Web Consortium (W3C) brings a new language to the Web as WebAssembly becomes a W3C Recommendation. In: *W3C* [online]. World Wide Web Consortium (W3C), 2022 [cit. 2023-04-11]. Dostupné z: https://www.w3.org/2019/12/pressrelease-wasm-rec.html.en

[31] DEVERIA, Alexis. Can I use: WebAssembly. In: *Caniuse* [online]. [cit. 2023-04-11]. Dostupné z: https://caniuse.com/wasm

[32] STAMENIC, Dimitrije. An Introductory Guide to Brython. In: *StackAbuse* [online]. Stack Abuse, © 2013-2023 [cit. 2023-04-10]. Dostupné z: https://stackabuse.com/an-introductory-guide-to-brython/

[33] Transcrypt. In: *GitHub* [online]. GitHub, Inc., 2023 [cit. 2023-04-10]. Dostupné z: https://github.com/qquick/Transcrypt

[34] Transcrypt: Python in the browser. In: *Transcrypt* [online]. [cit. 2023-04-10]. Dostupné z: https://www.transcrypt.org/home

[35] Getting started: Your first Transcrypt program. In: *Transcrypt* [online]. Jacques de Hooge, 2021 [cit. 2023-05-13]. Dostupné z: https://www.transcrypt.org/docs/html/installation_use.html#

[36] SHAUN. Running Python in the Web Browser. In: *Anvil: Build web apps with nothing but Python* [online]. Anvil, 2023 [cit. 2023-05-12]. Dostupné z: https://anvil.works/blog/python-in-the-browser-talk

[37] NIJBURG, Albert-Jan. Skulpt. In: *GitHub* [online]. GitHub, Inc., 2023 [cit. 2023-05-12]. Dostupné z: https://github.com/skulpt/skulpt

[38] Programming Skulpt: HOW TO. In: *Skulpt.org* [online]. [cit. 2023-05-12]. Dostupné z: https://skulpt.org/docs/index.html

[39] Using Skulpt: Using Skulpt with HTML. In: *Skulpt* [online]. 2015 [cit. 2023-05-12]. Dostupné z: https://skulpt.org/using.html

[40] PyPy.js: PyPy compiled into JavaScript: Using the Interpreter. In: *GitHub* [online]. GitHub, Inc., 2023 [cit. 2023-05-13]. Dostupné z: https://github.com/pypyjs/pypyjs/blob/master/README.dist.rst

[41] *Pyodide* [online]. Pyodide contributors and Mozilla, © 2019-2022 [cit. 2023-04-10]. Dostupné z: https://pyodide.org/en/stable/

[42] Pyodide 0.23.0 release. In: *Blog.pyodide.org* [online]. Pyodide contributors and Mozilla, 2023 [cit. 2023-05-13]. Dostupné z: https://blog.pyodide.org/posts/0.23-release/

[43] Getting started. In: *Pyodide* [online]. ©2019-2022: Pyodide contributors and Mozilla, ©2019-2022 [cit. 2023-05-13]. Dostupné z: https://pyodide.org/en/stable/usage/quickstart.html

[44] YANG, Sophia. PyScript: Python in the Browser. In: *Anaconda Nucleus* [online]. Anaconda Inc., 2023 [cit. 2023-05-13]. Dostupné z: https://anaconda.cloud/pyscript-python-in-the-browser

[45] Frequently asked questions. In: *PyScript - documentation* [online]. Anaconda, Inc, 2022 [cit. 2023-05-13]. Dostupné z: https://docs.pyscript.net/latest/reference/faq.html

[46] Element: Methods and Properties. In: *PyScript - documentation* [online]. Anaconda, Inc., 2022 [cit. 2023-05-09]. Dostupné z: https://docs.pyscript.net/latest/reference/API/element.html

[47] List of PyScript Attributes to Events. In: *PyScript - documentation* [online]. Anaconda, Inc., 2022 [cit. 2023-05-13]. Dostupné z: https://docs.pyscript.net/latest/reference/API/attr_to_event.html

[48] Packages built in Pyodide. In: *Pyodide* [online]. Pyodide contributors and Mozilla, ©2019-2022 [cit. 2023-05-13]. Dostupné z: https://pyodide.org/en/stable/usage/packages-in-pyodide.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PyPI	Python Package Index.
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
URL	Uniform Resource Locator
AJAX	Asynchronous JavsScript and XML
ECMA	European Computer Manufacturers Association
AST	Abstract syntax tree
API	Application Programming Interface
CDN	Content Delivery Network
DOM	Document Object Model
TOML	Tom's Obvious, Minimal Language
JSON	JavaScript Object Notation

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: PYTHON BALÍČKY DOSTUPNÉ V PYODIDE

Tabulka 1 – Python balíčky dostupné v Pyodide [48]

Název balíčku			
arduino	Copy code	asciitree	astropy
atomicwrites	attrs	autograd	bcrypt
beautifulsoup4	biopython	bitarray	bleach
bokeh	boost-histogram	brotli	cbor-diag
certifi	cffi	cffi_example	cftime
click	cligj	cloudpickle	cmyt
colorspacious	coverage	cramjam	cryptography
cssselect	cycler	cytoolz	decorator
demes	distlib	docutils	exceptiongroup
fastparquet	fiona	fonttools	freesasa
fsspec	future	galpy	gensim
geopandas	gmpy2	gsw	h5py
html5lib	idna	imageio	iniconfig
jedi	Jinja2	joblib	jsonschema
kiwisolver	lazy-object-proxy	lightgbm	logbook
lxml	MarkupSafe	matplotlib	matplotlib-pyodide
micropip	mne	more-itertools	mpmath
msgpack	msprime	multidict	munch
mypy	networkx	newick	nlopt
nltk	nose	numcodecs	numpy
opencv-python	optlang	packaging	pandas
parso	patsy	Pillow	pillow_heif

pkgconfig	pluggy	py	pyb2d
pyclipper	pycparser	pycryptodome	pydantic
pyerfa	Pygments	pyheif	pyinstrument
pynacl	pyodide-http	pyparsing	pyproj
pyrsistent	pytest	pytest-benchmark	python-dateutil
python-magic	python-sat	python_solvespace	pytz
pywavelets	pyxel	pyyaml	rebound
reboundx	regex	retrying	RobotRaconteur
ruamel.yaml	scikit-image	scikit-learn	scipy
setuptools	shapely	six	smart_open
soupsieve	sparseqr	sqlalchemy	statsmodels
svgwrite	swiglpk	sympy	tblib
termcolor	threadpoolctl	tomli	tomli-w
toolz	tqdm	traits	tskit
typing-extensions	uncertainties	unyt	webencodings
wordcloud	wrapt	xarray	xgboost
xlrd	yarl	yt	zarr

PŘÍLOHA P II: PŘILOŽENÉ CD

Součástí přiloženého CD k bakalářské práci je:

- bakalářská práce ve formátu PDF,
- zdrojové kódy ukázek použití frameworku PyScript.