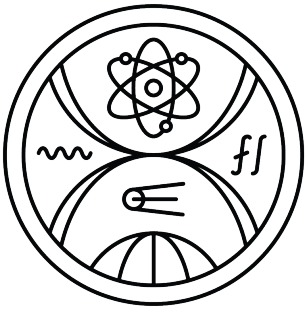
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

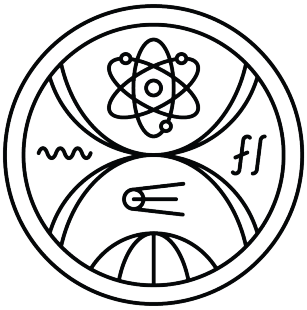


WEBASSEMBLY

Diplomová práca

2023 Bc. Klaudia Garajová

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



WEBASSEMBLY

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný obor:	Informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Mgr. Pavel Petrovič, PhD.

Bratislava, 2023 Bc. Klaudia Garajová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Klaudia Garajová
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Webassembly
Webassembly

Anotácia: Webassembly je assembler pre webový prehliadač. Dajú sa do neho kompilovať rôzne jazyky, pozri napríklad prehľad v [1].
Napriek tomu zostávajú aplikácie vo webovom prehliadači v štýle udalosťami riadeného programovania, zatiaľ čo pre mnohé iné jazyky platia iné paradigmy. Webassembly je alternatívou k inak jedinému podporovanému jazyku v prehliadačoch - Javascriptu, ktorý je interpretovaný a preto nie veľmi primeraný pre aplikácie, ktoré sú závislé na vysokej efektívnosti a výkone.

Cieľ: Cieľom tejto práce je vykonať dôkladnú analýzu technológie Webassembly, prakticky vyskúšať rozličné jazyky, ktoré sú kompilovateľné do Webassembly a och kompilátory, analyzovať ich výhody a obmedzenia a charakterizovať ich vo vzájomnom porovnaní. Okrem toho študent navrhne jednoduchý jazyk, ktorý bude kompilovaný do Webassembly a jeho programy budú bežať vo webových prehliadačoch. Ide teda prevažne o prehľadovú prácu nasledovanú praktickým originálnym implementačným projektom.

Literatúra:
[1] Paul Krill: 13 hot language projects riding WebAssembly, InfoWorld, April 2022, on-line: https://www.infoworld.com/article/3619608/13-hot-language-projects-riding-webassembly.html
[2] Mike Rourke: Learn Webassembly, Build Web Applications with Native Performance using Wasm and C/C++, Packt, 2018.
[3] Rick Batagline: The Art of Webassembly, Build Secure, Portable, High-Performance Applications, No Starch Press, 2021.
[4] Brian Sletten: WebAssembly, The Definitive Guide. Safe, Fast, and Portable Code, O'Reilly, 2021.

Kľúčové slová: webassembly, programovanie webu, vysoký výkon

Vedúci: Mgr. Pavel Petrovič, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 29.09.2022

Dátum schválenia: 03.12.2022

prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

.....
študent

.....
vedúci práce

Poďakovanie

Abstrakt

JavaScript bol dlhé roky dominantným programovacím jazykom používaným na webe, avšak s príchodom technológie nazývanej WebAssembly sa otvárajú dvere pre ďalšie možnosti. Cieľom tejto diplomovej práce je analyzovať technológiu WebAssembly a preskúmať jej možnosti využitia vo webovom prostredí. Práca navyše poskytne prehľad rôznych kompilátorov, ktoré umožňujú kompiláciu bežných programovacích jazykov do WebAssembly spolu s ich výhodami a obmedzeniami. Získané informácie sa v práci ďalej využívajú k navrhnutiu a implementácii vlastného jednoduchého programovacieho jazyka kompilovaného do WebAssembly. Na implementáciu sú použité technológie ANTLR4 a Binaryen JavaScript API. Funkčnosť vytvoreného jazyka je overená pomocou jednoduchých webových hier.

Kľúčové slová: WebAssembly, kompilátor, jednoduchý programovací jazyk

Abstract

JavaScript was for many years the dominate programming language on the web, but with the advent of technology called WebAssembly, new opportunities are opening up. The aim of the thesis is to analyse WebAssembly technology and explore its potential for use in a web environment. In addition, the thesis provides an overview of different compilers used to compile mainstream programming languages to WebAssembly along with their benefits and restrictions. Acquired knowledge is used to design and implement own simple programming language compiled to WebAssembly. ANTLR4 and Binaryen JavaScript API are used for the language implementation. Functionality of the created language is verified through simple web games.

Keywords: WebAssembly, compiler, simple programming language

Obsah

Kapitola 1

Úvod

[1] [2] [3] [4] [5] [6] [7]

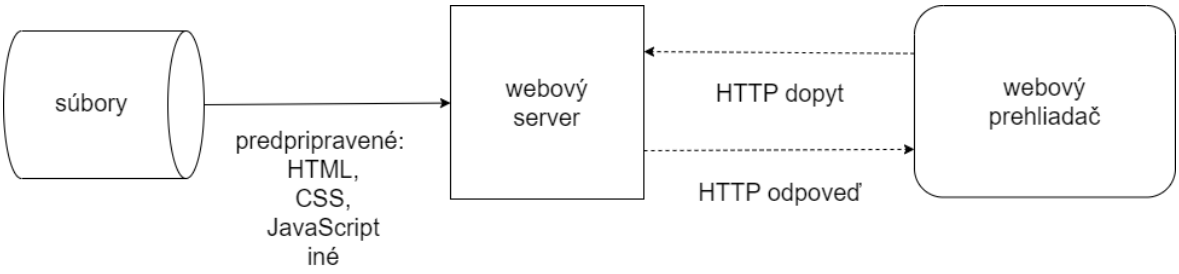
Kapitola 2

Webové stránky

Prvotná myšlienka webu vznikla v roku 1989. Britský informatik a vedec Tim Berners-Lee dostal nápad vytvoriť systém, ktorý by umožňoval zdieľať informácie medzi vedcami. O niekoľko rokov neskôr, konkrétne v roku 1991, vytvoril prvú webovú stránku, ktorá obsahovala informácie o tom čo je web a inštrukcie ako vytvoriť ďalšie stránky. Od toho momentu prešiel web obrovským vývojom. V tejto kapitole sa pozrieme ako sa web z pôvodného konceptu statických stránok postupne vyvinul do interaktívnych webových aplikácií s dynamickým obsahom. Pri skúmaní vývoja dynamického webu si zároveň predstavíme technológie, ktoré boli kľúčové pri jeho rozvoji.[8]

2.1 Statické webové stránky

Prvotné webové stránky boli statické. Všetkým svojim návštevníkom zobrazovali rovnaký obsah a poskytovali veľmi obmedzené možnosti interaktivity. Ich hlavným cieľom bolo sprístupnenie informácií používateľom bez geografických obmedzení. Aj mnohé súčasné webové stránky klasifikujeme ako statické. Ich obsah, štylizácia a dynamika sú uložené v HTML, CSS a JavaScript súboroch na webovom serveri. Webový server po obdržaní požiadavky poskytne webovému prehliadaču požadované súbory v nezmenenom stave. Teda ak by sme chceli zmeniť obsah statických webových stránok, museli by sme upraviť samotné súbory. [8][9]



Obr. 2.1: Proces získavania statických zdrojov

Statické webové stránky vďaka predpripravenému obsahu v HTML súboroch ponúkajú [10]:

- rýchle načítanie vo webovom prehliadači,
- nízku náchylnosť na nežiaducu manipuláciu dát,
- nízku hostingovú nákladnosť ako dôsledok nenáročného spracovávania na strane servera,
- ľahkú škálovateľnosť nevyžadujúcu ďalšie zdroje s nárastom používateľov.

Okrem spomenutých výhod môžeme pri statických weboých stránkach naraziť aj na nasledujúce obmedzenia: [10]:

- limitovaná interaktivita spôsobená obmedzenou funkcionalitou interaktívnych prvkov,
- absencia personalizovaného obsahu,
- časovo náročná aktualizácia obsahu v prípade rozsiahlych stránok či stránok s častými zmenami dát,
- obmedzená funkcionalita spôsobená neschopnosťou komunikovať so serverom a databázou.

Reakciou na obmedzenia statických webových stránok bol postupný vývoj dynamických webových stránok.

2.2 Dynamické webové stránky

Dynamická webová stránka, narozdiel od statickej webovej stránky, umožňuje zobrazenie rôzneho obsahu v závislosti od používateľa, časového pásma, času, historických dát prezerania a mnohých iných faktorov. Zároveň umožňuje aktualizovať obsah aj podľa akcií vykonaných používateľom. Všetky zmeny obsahu, prípadne zmeny layoutu sú možné vďaka generovaniu stránky v reálnom čase. [11]

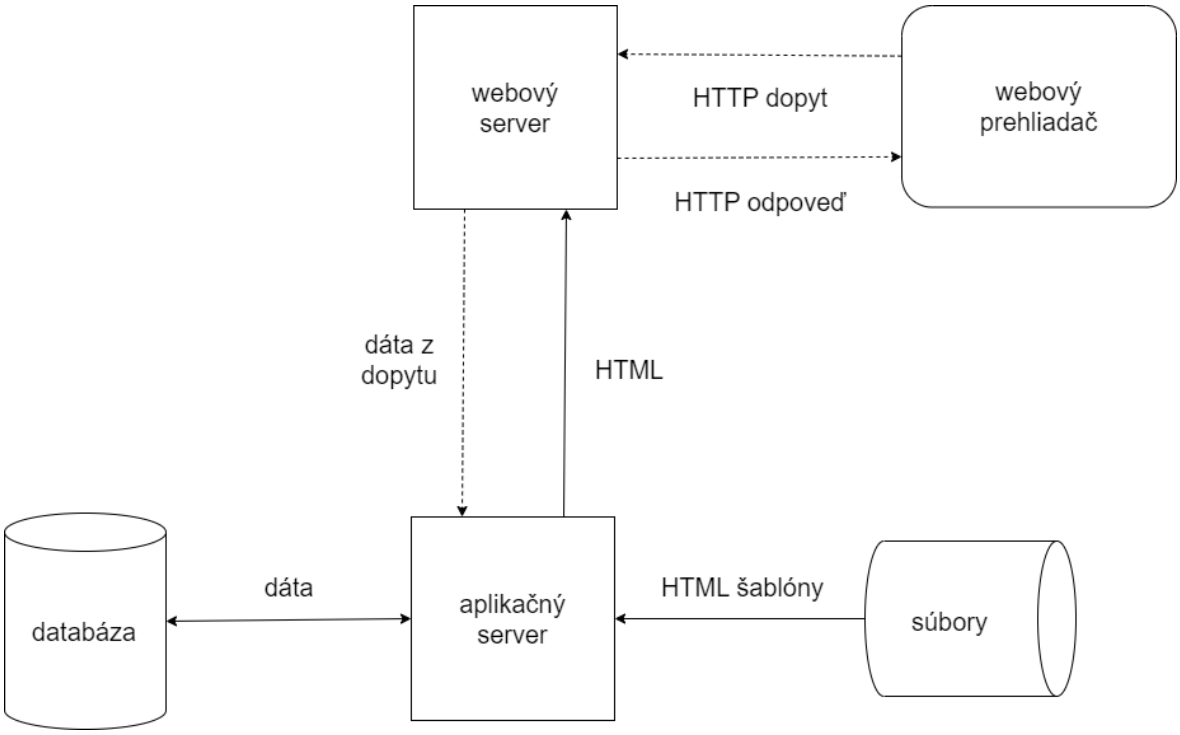
Samotný priebeh generovania závisí od typu dynamickej webovej stránky. Existujú dva základné typy: dynamické webové stránky generované na serveri (server-side dynamic web page) a dynamické webové stránky generované na klientovi (client-side dynamic web page). V súčasnej dobe sú veľmi časté dynamické webové stránky využívajúce kombináciu technológií používaných na strane servera s technológiami používanými na strane klienta. Na jednotlivé typy sa pozrieme detailnejšie v nasledujúcich podkapitolách. [9]

2.2.1 Dynamické webové stránky generované na serveri

Dynamická webová stránka generovaná na serveri umožňuje generovanie obsahu na základe rôznych vstupov, medzi ktoré sa radia napríklad parametre v URL dopyte, vstupy od používateľa, doménová logika alebo dáta z databázy, externých API či iných dátových zdrojov. Ako už samotný názov týchto stránok napovedá, dynamické generovanie obsahu sa deje na serveri, ktorý nazývame aplikačný server.

Celý proces vytvorenia dynamického obsahu prebieha nasledovne. Webový prehliadač pošle požiadavku na webový server, ktorý ju prepošle aplikačnému serveru. Aplikačný server v ďalšom kroku dotiahne predpripravenú HTML šablónu pre dopytovanú stránku. Šablóna je špeciálna tým, že navyše obsahuje kód v skriptovacom jazyku určeným na vývoj na strane servera (server-side scripting language). Daný kód môže vykonávať rôzne úlohy, ktoré môžu zahŕňať spracovanie vstupu od používateľa, manipuláciu údajov alebo dopytovanie do databázy či iných dátových zdrojov. V nasledujúcom kroku aplikačný server vykonáva kód a generuje HTML obsah, ktorý sa doplní do šablóny namiesto skriptovacieho kódu čím sa vytvorí statický HTML súbor. Na záver aplikačný server vygenerovaný statický HTML súbor pošle ako odpoveď webovému serveru, ktorý ho posunie webovému prehliadaču. Opísaný postup sa vykonáva s každým

novým dopytom webového prehliadača. [9][10][12]



Obr. 2.2: Generovanie dynamického obsahu na strane servera

Generovanie obsahu dynamickej webovej stránky na strane servera disponuje mnohými výhodami, medzi nimi napríklad [13]:

- možnosť komunikácie s databázou, resp. s inými zdrojmi,
- zabezpečenie dát vďaka ich spracovávaniu na serveri,
- ukrytie zdrojového kódu pred používateľom,
- rýchle načítanie stránky v prípade servera s dostatočnou výpočtovou silou.

Tiež spomenieme niektoré nevýhody, ktoré je potrebné si uvedomiť, aby sme predišli neželaným stavom[13]:

- spomalené načítanie stránky pri zložitých úlohách a nedostatočne výkonnom serveri,
- risk úniku dát pri slabo zabezpečenej stránke,
- vyššia latencia spôsobená častou komunikáciou medzi webovým prehliadačom a webovým serverom.

2.2.2 Prehľad technológii používaných na generovanie dynamického obsahu na serveri

V súčastnosti existuje mnoho rôznych technológií, ktoré sa používajú na generovanie dynamického obsahu na strane servera. Niektoré z týchto technológií existujú dlhšiu dobu, iné sú novšie. Niektoré sa pýšia rýchlosťou, iné bezpečnosťou a ďalšie jednoduchosťou. Avšak každá z nižšie uvedených technológií prispela novými možnosťami v oblasti vývoja dynamických serverových webových aplikácií.

CGI

Spoločné rozhranie sieťového priechodu (Common Gateway Interface), skrátene CGI, je protokol, ktorý zohral významnú úlohu v začiatkoch webového vývoja. Protokol umožnil komunikáciu medzi webovým serverom a externými skriptami, ktoré vďaka tomu dokázali generovať dynamický obsah podľa dopytov webového prehliadača. Bohužiaľ, každý dopyt predstavoval nový proces, čo pri väčšej premávke mohlo viesť k preťaženiu systémových zdrojov. [9][12]

ASP

Active Server Pages, skrátene ASP, je modul od Microsoftu, ktorý sa pripája ku webovému serveru a slúži na spracovanie skriptov ako aj ich vykonanie. ASP podporuje rôzne skriptovacie jazyky, predovšetkým VBScrip a JScript. Skriptovací kód sa nachádza spolu s HTML kódom v tzv. ASP súbore. Webový server po obdržaní dopytu na ASP súbor, prepošle dopyt na ASP modul, ktorý vykoná skriptovací kód pre danú stránku. Výsledkom je HTML súbor, ktorý sa pošle ako odpoveď webovému prehliadaču. Nevýhodou ASP je istá platformová závislosť, kedže tento modul bol vyvinutý primárne pre platformu Windows. Okrem toho zložitejšie výpočty či väčšia premávka výrazne spomaľujú výkon. [12]

PHP

PHP je programovací jazyk, ktorý vznikol v roku 1994 a od tej doby si prešiel mnohými vylepšeniami. Patrí medzi skriptovacie jazyky využívané na strane servera. Aby server mohol podporovať PHP jazyk musí obsahovať jeho interpretátor, ktorý po obdržaní dopytu na PHP stránku vykonáva patričný PHP kód a zároveň generuje výstup pre

klienta. PHP kód je zvyčajne integrovaný priamo do HTML kódu pomocou tagov. Nad PHP jazykom vznikli mnohé frameworky, ktoré uľahčujú a zrýchľujú vývoj webových stránok. Medzi najznámejšie patrí Laravel alebo Symphony. [11][13]

Servlet

Servlet je trieda programovacieho jazyka Java, ktorá dokáže spracovať dopyt klienta a vygenerovať odpoveď. Podobne ako CGI, servlet predstavuje isté prepojenie medzi klientom a databázou či rôznymi aplikáciami. Životný cyklus tejto triedy má na starosti Servlet kontajner, ktorý je súčasťou webového prípadne aplikačného servera. Práve Servlet kontajner vďaka konfigurácii vie pre každý dopyt nainicializovať a rozbehnúť správny servlet.[9]

JSP

JSP, Java Server Pages, je technológia postavená na Servletoch s cieľom uľahčiť tvorbu webových stránok. Zatiaľ čo Servlety umožňovali generovať dynamický obsah v Java kóde pomocou print metód, JSP technológia používa JSP stránky, ktoré umožňujú písať kód v programovacom jazyku Java priamo do HTML štruktúry, a tak dosiahnúť dynamický obsah. Pri dopyte na JSP stránku, softvér nazývaný JSP engine vytvorí podľa kódu v JSP stránke Servlet, ktorý následne vykonáva daný kód a generuje HTML stránku.[9][12]

JakartaEE

JakartaEE, predtým známa aj ako JavaEE, je rozšírenie platformy Java o funkcionality, ktoré využívajú vývojári vo veľkých a komplexných aplikáciách v podnikoch. Táto aplikačná softvérová platforma obsahuje veľké množstvo API, ktoré umožňujú vývojárom používať pokročilé funkcie. Vďaka ním môžu vývojári vytvárať výkonné aplikácie, ktoré plnia náročné požiadavky z podnikového prostredia, napr. bankové informačné systémy, e-commerce, správu skladov, účtovníctvo a iné.Medzi hlavné prvky JavaEE zaraďujeme Java Beans(EJB), Java Server Pages(JSP) a Java Servlety a iné ďalšie rozhrania, ktoré sa využívajú na prepájanie so zdrojmi v podniku.[14]

ASP.NET

ASP.NET je framework vytvorený firmou Microsoft ako vylepšený nástupca ASP modulu. Vylepšenie možno spozorovať v rozšírení škály podporovaných jazykov. ASP.NET nezostal len pri skriptovacích jazykoch, ale rozšíril podporu pre rôzne jazyky podporované .NET platformou. S touto zmenou súvisí aj pridaná podpora objektovo-orientovaného programovania či modularizácie aplikácie. ASP.NET navyše ponúka lepší výkon, lepšiu bezpečnosť a bohatú podporu webových štandardov.[12]

Ruby on Rails

Ruby on rails, skrátene sa označuje ako RoR alebo Rails, je webový framework napísaný v jazyku Ruby. Rails je jednoduchý, rýchly framework, ktorý vývojári využívajú pri vytváraní robustných webových aplikácií. K zjednodušeniu vývoja prispieva aj veľké množstvo knižníc, ktoré Rails obsahuje. Jedná sa o nástroje na migráciu databázy, správu relačných databáz, rýchle prototypovanie a ďalšie. Vývojárom napomáha aj skutočnosť, že základné funkcie nie je potrebné opakovane implementovať. Tento framework je používaný v rôznych aplikáciách, napr. e-commerce, sociálnych sieťach a iných.[15]

Django

Django je framework, ktorý sa používa na strane servera na rýchlu tvorbu komplexných webových aplikácii, ktoré sa navyše charakterizujú bezpečnosťou a škálovateľnosťou. Framework podporuje princíp DRY (Don't Repeat Yourself), ktorého snahou je odstránenie duplicitného kódu. Podporuje šablóny, ORM (Object-Relational Mapping), pohľady, znovupoužiteľné formuláre a mnohé iné.[16]

Node.js

Node.js je framework, ktorý umožňuje využitie programovacie jazyka JavaScript aj na generovanie obsahu na strane servera. Využíva asynchronné spracovanie udalostí, vďaka čomu je schopný spracovať efektívne veľké množstvo súbežných požiadaviek. Okrem toho veľkou výhodou tohto frameworku je možnosť využitia existujúcich modulov ponúkajúcich rôzne funkcionality. Tieto moduly sú k dispozícii v centrálnom repozitári nazývaném npm (Node Package Manager) odkiaľ ich je možné veľmi jednoducho

stiahnuť a nainštalovať do aplikácie.[17][18]

Spring Boot

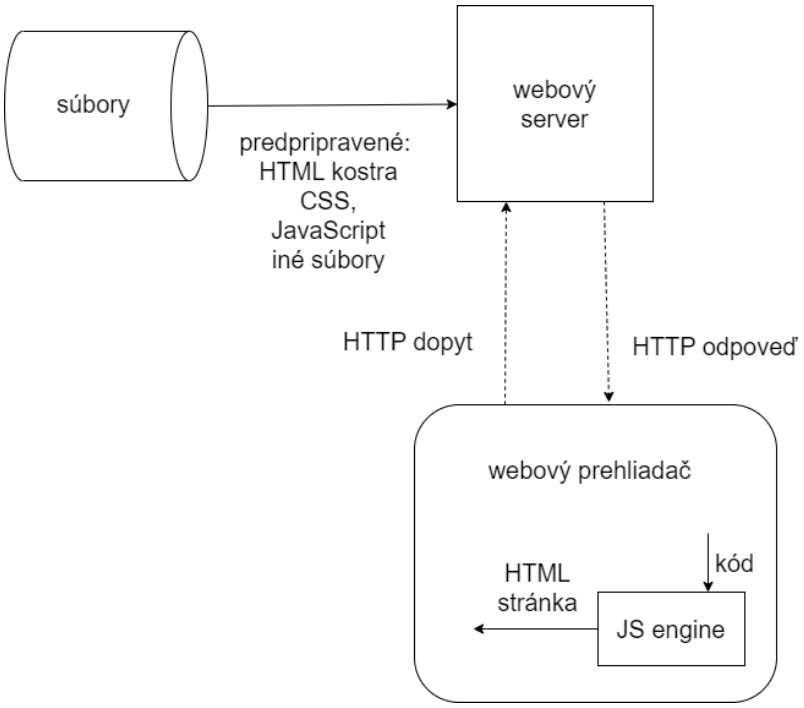
Spring Boot je framework slúžiaci na vývoj aplikácii v jazyku Java. Aj tento framework umožňuje rýchle a jednoduché vytváranie aplikácií. Poskytuje automatickú konfiguráciu a množstvo rôznych nástrojov, vďaka čomu sa programátori môžu pri vývoji aplikácie sústrediť viac na funkcionalitu danej aplikácie než na zdĺhavé manuálne nastavovania a konfigurácie.[19]

2.2.3 Dynamické webové stránky generované na klientovi

Dynamické webové stránky generované na klientovi generujú obsah stránky na klientovi, teda vo webovom prehliadači. Inými slovami, doťahovanie dát, vykonávanie biznis logiky, rútovanie a ďalšie iné úlohy sa vykonávajú priamo vo webovom prehliadači. Aby sa všetky tieto úlohy mohli vykonať, najskôr sa pošle požiadavka webovému serveru na požadovanú stránku. Následne webový server vráti webovému prehliadaču HTML súbor. Tento súbor väčšinou v skutočnosti obsahuje len kostru stránky s prípadným načítavačom (loader). Webový server okrem toho vráti aj potrebné CSS súbory a súbory obsahujúce potrebný skriptovací kód. Webový prehliadač načíta HTML súbor, zparsuje ho a vytvorí DOM, stromovú reprezentáciu dokumentu. Tiež načíta CSS súbor a aplikuje dané štýly na jednotlivé uzly vo vytvorenom DOM strome. Dôležitá časť prichádza po načítaní skriptovacieho kódu. Webový prehliadač začne vykonávať daný kód. Keďže kód má prístup k DOM stromu, môže ho rôzne upravovať, pridávať či odoberať uzly, upravovať štýly, doťahovať potrebné dáta asynchrónnym spôsobom a tiež spracovávať používateľské interakcie. Výsledkom opísaného procesu je viditeľná dynamická interaktívna webová stránka. [9][20]

Medzi benefity, ktoré nám priniesli dynamické webové stránky generované na klientovi určite patrí [20]:

- vysoká interaktivita a responzívnosť,
- nižšie náklady na hosting vďaka nižšiemu zaťaženie servera,
- rýchle načítanie stránok (zväčša okrem prvotného dopytu).



Obr. 2.3: Generovanie dynamického obsahu na strane klienta

Pri dynamickom generovaní obsahu na klientovi môžeme naraziť na nasledujúce problémy [20]:

- dlhšie prvotné načítanie stránky kvôli potrebe stiahnutia celého súboru so skriptovacím kódom a následného renderovania obsahu,
- závislosť výsledného používateľského zážitku od výkonu používateľovho zariadenia a webového prehliadača,
- bezpečnostné riziko spôsobené spracovávaním údajov priamo vo webovom prehliadači,
- znížená viditeľnosť stránky vo vyhľadávaní kvôli problémom s indexáciu dynamicky generovaného obsahu.

Zatiaľ čo serverové dynamické webové stránky často ukladajú dáta v databázach alebo externých úložiskách, klientske webové stránky nám dovoľujú ukladať dočasné údaje priamo na zariadení používateľa. Lokálnemu ukladanie dát spôsobuje rýchlejšie načítanie webových stránok a promptnejšie spracovanie dát. Dáta nie je potrebné opakovane načítavať zo servera, teda rýchlosť odozvy sa zvyšuje a zároveň prechádzanie medzi rôznymi webstránkami sa zrýchli. Takéto zlepšenie výkonu webovej stránky

súčasne zvyšuje pozitívnu používateľskú skúsenosť. Výhodou aj je ukladanie citlivých používateľských údajov na ich zariadenia. Používatelia sa sami tiež môžu rozhodnúť, ktoré dáta sa nebudú ukladať alebo budú odstránené, čím je zabezpečená aj používateľská kontrola údajov. Ďalšou výhodou je offline funkcionalita, teda užívatelia môžu používať webové aplikácie aj keď nie sú pripojení na internet alebo chcú využiť výhodu offline režimu. Tiež spomenieme zníženie zaťaženia servera, keďže nie je potrebné dopytovať údaje pri každom prístupe na stránku. Všetky tieto výhody umožňujú väčšiu flexibilitu, rýchlosť a efektívnosť v interakcii s webovými aplikáciami. V súčastnosti existujú nasledujúce typy ukladania dát na strane klienta.[21]

Web Storage

Webové úložisko (Web Storage) je API, ktoré umožňuje webovým aplikáciám uchovávať dáta priamo v prehliadači užívateľa. V rámci webového úložiska rozlišujeme dva mechanizmy.

- Lokálne úložisko (localStorage), ktoré ukladá dáta na neobmedzenú dobu. Dáta v lokálnom úložisku nemajú nastavenú expiračnú dobu, ale sú stále dostupné aj po znovuotvorení webstránok.
- Úložisko relácie (sessionStorage), ktoré umožňuje ukladanie dát iba počas aktuálnej relácie prehliadača. Akonáhle sa okno webstránky zatvorí, prípadne používateľ prejde na inú stránku, tak sa dáta vymažú. Výhodou krátkodobého úložiska je ukladanie krátkodobých informácií, napr. stav nákupného košíka.

Práca s Web Storage prebieha pomocou JavaScriptu cez jednoduché API, pričom dáta sa ukladajú v pároch kľúč-hodnota.[21]

Cookies Cookies môžeme definovať ako malé textové súbory, ktorých úlohou je ukladať informácie v prehliadači na strane klienta. Cookies ukladajú používateľské dáta, ktoré môžu byť neskôr využité na personalizovaný obsah či sledovanie aktivít. Rozlišujeme nasledujúce dva typy.

- Session Cookies sú dočasné cookies, ktoré ukladajú dáta v používateľskom prehliadači len počas relácie na webstránke. Cookies sa zvyčajne vymazávajú hneď po zatvorení okna prehliadača.

- Persistent Cookies, čiže trvalé cookies ukladajú dáta v prehliadači používateľa určité obdobie a po uplynutí stanoveného obdobia sú vymazané.

Veľkosť cookies je obmedzená, a teda cookies nie sú obmedzené len dĺžkou trvania, ale aj množstvom informácií.[21]

IndexedDB

IndexedDB je API slúžiace ukladanie väčšieho množstvo dát na strane klienta. Toto úložisko funguje na pricípe databázy, teda je nad ním možné vykonávať všetky základné operácie, ako ukladanie, čítanie, upravovanie a vyhľadávanie dát. Veľkou výhodou je, že dané úložisko nepotrebuje permanentné pripojenie na internet a umožňuje efektívne vyhľadanie a spracovanie aj nad väčším množstvom uložených dát.[21]

Cache API

Cache API umožňuje ukladanie dát do Cache úložiska, ktoré sa nachádza v prehliadači a je oddelené od ostatných. Dáta sa ukladajú v páre kľúč-hodnota. Kľúč zvykne byť URL adresa a hodnotou sú dáta, ktoré chceme uložiť. Cache úložisko sa najčastejšie využíva na ukladanie statických dát, napr. obrázky, CSS či JavaScript súbory. Vďaka uloženiu do tohto úložiska sa webové stránky vedia pomerne rýchlo načítať. Toto úložisko môžeme využiť aj na ukladanie dát potrebných pri používaní stránky v offline režime.[21]

File system storage

File system storage je experimentálne API, ktoré poskytuje prístup k súborom, a teda dovoľuje ich vytvárania, čítanie, zapisovanie, ale aj mazanie. Okrem toho umožňuje istú správu súborového systému pozostávajúcu z vytvárania adresárov, presúvania súborov do rôznych adresárov ako aj kopírovanie súborov. Tento typ ukladania je najmä výhodný pre veľké dát akými sú napr. obrázky, videá či zvukové záznamy. Veľkou nevýhodou je, že prístup k súborom môže byť zneužitý na získanie citlivých dát, zároveň ak dôjde ku chybe používateľ môže prísť o vlastné dáta.[21]

2.2.4 Prehľad technológii používaných na generovanie dynamického obsahu na klientovi

Rast a pokrok na klientovej strane webu priniesol mnohé technológie, ktoré umožnili interaktívne dynamické webové stránky. Niektoré kvôli nevýhodám stratili popularitu medzi vývojármi iné stratili podporu v moderných prehliadačoch. Ich vplyv na používateľskú interakciu na webových stránkach je ale dôležitý.

Java Applet

Java Applet je staršia technológia, ktorá v súčastnosti už nie je podporovaná prehliadačmi. Java applet je program napísaný v programovacom jazyku Java, prípadne inom programovacom jazyku, ktorý sa kompiluje to Java bajtkódu. Tieto programy sa mohli priamo vložiť do html kódu pomocou špeciálnej značky, ktorá obsahovala informáciu o ceste k súboru s daným bajtkódom. Akonáhle používateľ navštívil stránku, ktorá obsahovala applet, webový prehliadač stiahol bajtkód z danej cesty a spustil ho v rámci svojho Java virtuálneho stroja (JVM). Applet reagoval na udalosti od používateľa a vytváral vizuálny obsah ako je text či animácie. Zároveň Java Applet bežal v sandboxe, vďaka čomu stiahnutý kód mal obmedzený prístup k systému a systémovým zdrojom. Avšak tento čiastočný prístup bol mnohokrát využitý ako brána na infikovanie počítača používateľa čo spolu s potrebou nainštalovania Java Runtime Environmentu(JRE) viedlo k postupnému úpadku tejto technológie.[12]

JavaScript

JavaScript je interpretovaný programovací jazyk vyvinutý spoločnosťou Netscape. Slúži na tvorbu interaktívnych webových stránok či aplikácií. JavaScript kód sa vykonáva priamo vo webovom prehliadači čo umožňuje okamžité reakcie na používateľské akcie. V súčastnosti je JavaScript podporovaný vo väčšine moderných webových prehliadačov. Navyše nad Javascriptom sú postavené moderné frameworky ako React, Angular, Vue.js, ktoré rozširujú možnosti Javascriptu a zjednodušujú vývoj moderných webových aplikácií. Javascript je kľúčový jazyk pre interaktívne dynamické webové stránky, a preto ho podrobne preskúmame v nasledujúcej kapitole.[22]

VBScript

VBScript je skriptovací jazyk, ktorý vyvinula spoločnosť Microsoft. Stavia na programovacom jazyku Visual Basic. Vo webových prehliadačoch slúžil na vytváranie interaktívneho obsahu, tiež na reagovanie na používateľské akcie, a teda môžeme povedať, že z hľadiska funkcionality bol priamym súperom JavaScriptu. Najväčšou nevýhodou bola podpora obmedzená len na webový prehliadač Internet Explorer, a tak postupom času sa tento skriptovací jazyk prestal používať na vývoj webových stránok.[12]

JScript

JScript je, podobne ako VBScript, skriptovací jazyk vyvinutý spoločnosťou Microsoft. Jeho syntax je veľmi podobná JavaScriptu, nakoľko tento jazyk vznikol ako Microsoftová implementácia jazyka JavaScript pre webový prehliadač Internet Explorer. S inými prehliadačmi nebol kompatibilný a nemal plnú podporu štandardov, preto postupom času vývojári preferovali štandardnejší JavaScript.[23]

TypeScript

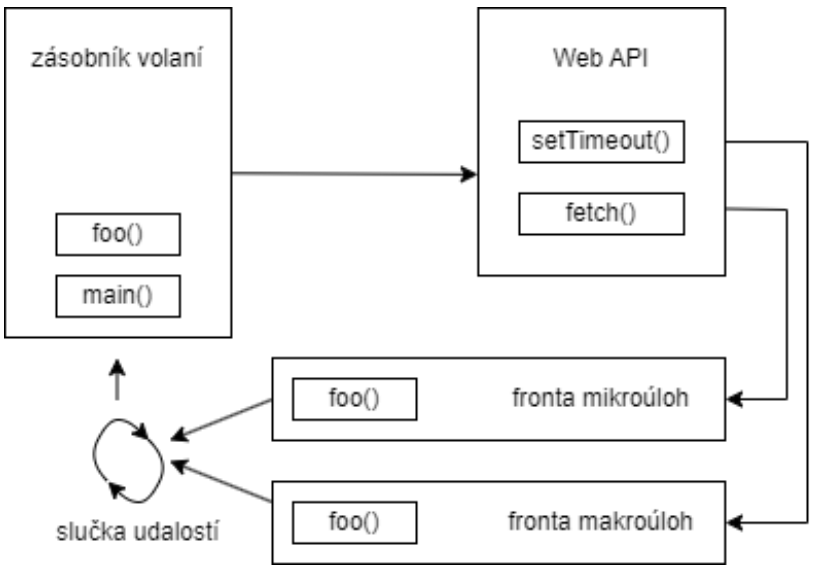
TypeScript je programovací jazyk, ktorý je nadstavbou jazyka JavaScript. Medzi pridanou funkcionalitou je statické typovanie, teda JavaScript umožňuje definovať typy čím predchádza typovým chybám v kóde. S tým je spojená aj typová inferencia, ktorá dokáže automaticky odvodiť typy z kódu. Pred samotným spustením kódu sa kód preloží z TypeScriptu do JavaScriptu pričom sa kontroluje správnosť typov. Veľkou výhodou je, že je teda kompatibilný s JavaScript kódom. Vďaka týmto výhodám sa stáva čoraz viac obľubeným jazykom na vývoj moderných webových aplikácií.[24]

2.2.5 JavaScript

JavaScript je skriptovací programovací jazyk, ktorý zohráva kľúčovú úlohu vo vývoji moderných webových aplikácií. Hlbšie pochopenie niektorých podstatných aspektov JavaScriptu nám umožňuje presnejšie využiť jeho potenciál a jeho možnosti integrácie s ďalšími technológiami. Súčasne nám pomôže pripraviť sa na nové možnosti, ktoré nám prinesie spolupráca s WebAssembly v nasledujúcej kapitole.

Jedným z dôležitých aspektov JavaScriptu je asynchrónnosť, keďže práve vďaka nej vieme udržať interaktívnosť na stránke. Samotný JavaScript je jednovláknový, čo

znamená, že kód sa môže vykonávať iba sekvenčne a synchronne. Ak by JavaScript neposkytoval aj asynchrónny model, ľahko by sa stalo, že stránka by sa pri časovo náročnom procese zasekla a používateľ by s ňou nemohol interagovať pokiaľ by daný proces neskončil. Základom Javascriptového asynchrónneho modelu je slučka udalostí (event loop), ktorá neustále kontroluje stav zásobníka volaní (call stack), fronta makroúloh (macrotask queue) a fronty mikroúloh (microtask queue). Zásobník volaní je zásobník, do ktorého sa na vrchol umiestni volanie vykonávanej funkcie, keď je funkcia skončená, odstráni sa aj z vrcholu zásobníka. Keď nastane nejaká udalosť ako napríklad spustenie setTimeout či setInterval alebo GUI udalosť, jej príslušný callback alebo event handler sa vloží do tejto fronty makroúloh pre budúce vykonanie. Naopak callbacky z promisov sa vkladajú do fronty mikroúloh. Slučka udalostí sleduje zásobník volaní, ak je tento prázdny vyberie nasledujúci callback z fronty mikroúloch a pridá ho do zásobníka na vykonanie. V poradí v akom callbacky vložené do fronty sa postupne všetky vykonajú. Ak je fronta mikroúloh prázdna začnú sa vykonávať callbacky z fronty makroúloh. Vďaka tomuto procesu stránka v JavaScripte zostáva interaktívna a responzívna.[22]



Obr. 2.4: JavaScriptový asynchrónny model

Ďalším mechanizmom ako zabrániť blokovaniu hlavného vlákna a umožniť paralelné spracovanie je Web Worker, ktorý umožňuje spúštanie kódu v oddelenom vlákne. Zároveň nemá prístup k DOM-u ani k hlavnému UI vláknu čím sa zabraňuje nežiaducemu blokovaniu používateľského rozhrania. Web Workeri môžu pomocou Web API

komunikovať s rôznymi službami či zdrojmi dát pričom komunikácia môže byť synchronna alebo asynchrónna. Tiež môžu asynchrónne komunikovať s hlavným vláknom, a to nasledovne. V hlavnom vlákne sa vytvorí referencia na daný Web Worker a následne pomocou metódy postMessage sa mu môže poslať správu z daného vlákna. Správa môže byť obyčajná textová správa, ale aj správa pozostávajúca zo štrukturovaných údajov, ako napríklad objekt či pole. Web Worker vie pomocou onmessage event listenera zachytiť správu a podľa potreby ju spracovať. Komunikácia opačným smerom prebieha obdobne. Web Worker môže poslať správu hlavnému vláknu pomocou metódy postMessage a dané vlákno ju vie odchytiť pomocou onmessage event listenera. Implementácia týchto metód a event listenerov je súčasťou samotného prehliadača. Avšak existuje špecifikácia pre Web Workre a ich API, a teda webové prehliadače implementujú toto API podľa špecifikácia.[22]

Na rozšírenie funkcionality webových prehliadačov sa v súčasnosti používa Web API, pred ním sa však používali rôzne pluginy, s ktorými sa dalo komunikovať práve cez JavaScript. Pluginy sú programy, ktoré boli vytvorené pomocou rôznych programovacích jazykov a ich implementácia bola väčšinou špecifická pre daný prehliadač. Rozširovali možnosti prehliadača napríklad o možnosť prehrávania multimédií, o možnosť interakcie so súborom so špecifickým formátom ale tiež poskytovali rôzne ďalšie užitočné nástroje pre vývojárov. Integrovať ich do webovej stránky bolo možné pomocou špeciálnych značiek HTML. Pluginy teda umožnili rýchlu integráciu rôznej funkcionality do webových stránok. Veľkou nevýhodou pluginov však bola ich potrebná inštalácia, ktorá mohla byť pre používateľov zložitá. Zároveň, aby sa správne zobrazoval obsah stránky, bolo potrebné mať aktuálnu verziu daného pluginu. Veľkým problémom bola aj samotná bezpečnosť, nakoľko pluginy mohli mať bezpečnostné diery, ktoré mohli byť využité na útoky používateľovho systému. Kvôli týmto podstatným nedostatkom sa pluginy postupne prestali používať a prešlo sa na modernejšie štandardy, medzi ktoré patrí aj už spomínané Web API.[25]

Web API vzniklo s cieľom umožniť webovým aplikáciám pristupovať k funkcionalite či dátam iných systémov, a tak rozšíriť funkcionalitu danej webovej aplikácie. Web API je sada pravidiel a protokolov, ktoré umožňujú komunikáciu cez internet medzi webovou aplikáciou a iným systémom či službou. Aplikácia posiela požiadavky na URL adresy pomocou štandardných protokolov ako je HTTP alebo WebSocket. Dáta v tejto

komunikáciu sú v štruktúrovanom formáte, najčastejšie v JSON alebo XML formáte. Web API môžeme rozdeliť na dve základné skupiny, verejné a súkromné. Už ako názvy nám napovedajú, verejné API je API, ktoré je dostupné pre verejnosť. Naopak súkromné API často vyžaduje autorizovaný prístup a zvyčajne sú používané napríklad vrámci organizácie. Teda medzi výhody API patrí zabezpečenie, nakoľko môže vyžadovať autentifikáciu a autorizáciu. Sú štandardizované, a teda umožňujú jednotný spôsob interakcie s rôznymi aplikáciami. Medzi nevýhody by sme mohli zaradiť obmedzenú výkonnosť pri väčšom počte požiadaviek. Niektoré API môžu byť zložitejšie na pochopenie ich funkcionality, avšak väčšinou majú slušnú dokumentáciu na rýchlejšie zorientovanie.[26]

Kapitola 3

WebAssembly

3.1 Motivácia pre WebAssembly

3.2 Technológia WebAssembly

WebAssembly, skrátene Wasm, je binárny kód, ktorý je vykonávaný pomocou Wasm virtuálneho stroja. Tento stroj používa zásobník na ukladanie a spracovanie dát. Wasm poskytuje abstraktnú vrstvu medzi programovacími jazykmi a konkrétnymi cieľovými platformami. To umožňuje programátorom písať kód v rôznych programovacích jazykoch, ako napríklad C, C++, Rust, a preložiť ho do WebAssembly. Následne tento Wasm kód môže byť spustený vo webovom prehliadači alebo na serveri pomocou Wasm virtuálneho stroja.[6]

Webasssemly je binárny formát avšak existuje aj textový formát nazývaný Wat, teda Wat je ako assemblerový jazyk pre Wasm virtuálny stroj.

```
(module
    (func (export "Add_Int")
    (param $value_1 i32) (param $value_2 i32)
    (result i32)
        local.get $value_1
        local.get $value_2
        i32.add
    )
)
```

Obr. 3.1: WebAssembly textový formát Wat

3.3 Interakcia s webovým prostredím

3.3.1 Integrácia modulu

3.3.2 DOM API

Vo WebAssembly 1.0, aktuálnej podporovanej verzii všetkými veľkými prehliadačmi, nevieme priamo komunikovať s DOM API z Wasm modulu. Keďže nemáme prístup k DOM API nevieme priamo meniť DOM štruktúru. Po skúmaní momentálnych možností Wasm sme usúdili, že jedinou možnosťou ako interagovať s DOM API je cez nepriamu interakciu. Teda pri inštanciovaní modulu v JavaScripte, si v rámci importovaného objektu pošleme do Wasm modulu JavaScriptové funkcie, ktoré volajú DOM API. Do budúcna sa plánuje pridať možnosť referencovať DOM a aj iné Web API objekty priamo z Wasm modulu, avšak zatiaľ je to stále vo fáze návrhu.[27][28]

3.3.3 Canvas API

Podobne ako DOM API, ani Canvas API nie je podporované v aktuálnej verzii WebAssembly. Ak napriek tomu chceme komunikovať a vykresľovať dáta do canvasu, máme nasledujúce 2 možnosti. Prvá možnosť je obdobná ako pre komunikáciu s DOM API, a teda môžeme využiť nepriamu komunikáciu pomocou JavaScriptových funkcií opísanú v podkapitole DOM API. Avšak v prípade, že chceme len vykresľovať dáta do canvasu môžeme využiť možnosť zdieľania pamäte WebAssembly s JavaScriptom, a teda dáta uložené v lineárnej WebAssembly pamäti môžeme priamo meniť vo Wasm module a následne už len JavaScript, ktorý má prístup k danej pamäti renderuje tieto dáta do Canvasu. Takýto prístup je veľmi užitočný v prípade vykresľovania animácií, kedy náročné operácie sú vykonávané efektívne vo Wasm module a JavaScript sa už len stará o renderovanie dát do canvasu.[27]

3.3.4 Web Workers

3.4 Komplexný príklad a optimalizácie

Kapitola 4

Jazyky a prostredia kompilované do WebAssembly

4.1 C/C++

4.2 Java

4.3 C#

4.4 Python

4.5 Go

4.6 Rust

4.7 AssemblyScript

4.8 Exotické jazyky

Kapitola 5

Návrh a implementácia jednoduchého programovacieho jazyka

Kapitola 6

Výzvy do budúcnosti

Záver

Literatúra

[1] Weihang Wang. Empowering web applications with webassembly: Are we there yet? *IEEE*, 2021. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9678831. Posledný prístup: 11.11.2023.

[2] Lijian Zhao a kol. Yutian Yan, Tengfei Tu. Understanding the performance of webassembly applications. *ACM*, 2021. https://weihang-wang.github.io/papers/imc21.pdf. Posledný prístup: 11.11.2023.

[3] Lennard Golsch. Webassembly: Basics. https://www.lennard-golsch.de/article/webassembly.pdf. Posledný prístup: 11.05.2023.

[4] Derek L. Schuff a kol. Andreas Haas, Andreas Rossberg. Bringing the web up to speed with webassembly. *ACM*, 52(6):185–200, 2017.

[5] Arjun Guha a kol. Luna Phipps-Costin, Andreas Rossberg. Continuing webassembly with effect handlers. *ACM*, 7(2):460–485, 2023.

[6] Rick Battagline. *The Art of WebAssembly, Build Secure, Portable, High-Performance Applications*. No Starch Press, 2021.

[7] Brian Sletten. *WebAssembly: The Definitive Guide: Safe, Fast, and Portable Code*. O'Reilly Media, 2021.

[8] Hadi Khosravi Farsani Sareh Aghaei, Mohammad Ali Nematbakhsh. Evolution of the world wide web: From web 1.0 to web 4.0. *International Journal of Web & Semantic Technology*, 3, 2012. https://airccse.org/journal/ijwest/papers/3112ijwest01.pdf. Posledný prístup: 25.11.2023.

[9] P. Karthik. *Web Application using JSP (Java Server Page)*. BPB Publications, 2020.

[10] R. Manjunath. *C, C++, Java, Python, PHP, JavaScript and Linux for Beginners.* Manjunath.R, 2019.

[11] Sham Tickoo. *Introducing PHP 7/MySQL.* CADCIM Technologies, 2018.

[12] Matt Butler a kol. Rob Birdwell. *Beginning ASP.NET 1.0 with C#.* Wrox, 2002.

[13] Tom Butler. *PHP & MySQL: Novice to Ninja.* SitePoin, 2022.

[14] Rhuan Rocha. *Jakarta EE for Java Developers.* BPB Publications, 2021.

[15] Ruby on rails guides. `https://guides.rubyonrails.org`. Accessed on: 2.12.2023.

[16] Arun Ravindran Samuel Dauzon, Aidas Bendoraitis. *Django: Web Development with Python.* Packt Publishing, 2016.

[17] Node.js dokumentácia. `https://nodejs.org/en/docs`. Accessed on: 3.12.2023.

[18] Arun Ravindran Samuel Dauzon, Aidas Bendoraitis. *Node.js Guidebook.* Packt Publishing, 2016.

[19] Spring boot reference documentation. `https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/`. Accessed on: 3.12.2023.

[20] Srikant Sahoo. *The Complete System Design for Frontend Developers.* nezávisle publikované, 2023.

[21] Client-side storage. `https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Client-side_storage`. Accessed on: 27.11.2023.

[22] Luis Atencio. *The Joy of JavaScript.* Manning Publications Co., 2021.

[23] Jscript(ecmascript3). `https://learn.microsoft.com/en-us/previous-versions//hbxc2t98(v=vs.85)?redirectedfrom=MSDN`. Accessed on: 1.12.2023.

[24] Stefan Baumgartner. *TypeScript Cookbook.* O'Reilly Media, 2023.

[25] What is the difference between api and plugin? https://incora.software/insights/what-is-the-difference-between-api-and-plugin. Accessed on: 27.11.2023.

[26] What is web api and why we use it ? https://www.geeksforgeeks.org/what-is-web-api-and-why-we-use-it/. Accessed on: 27.11.2023.

[27] Webassembly specification. https://webassembly.github.io/spec/core/. Accessed on: 5.12.2023.

[28] Webassembly proposals. https://github.com/WebAssembly/proposals. Accessed on: 5.10.2023.