Node.js este un mediul de rulare a limbajului de programare JavaScript, care este construit pe baza Chrome V8 Engine, acesta din urma fiind la randul lui un motor de inalta performanta menit pentru codul scris in JavaScript si WebAssembly. V8 este scris in C++, poate fi de sine statator sau poate fi integrat in orice aplicatie C++.

Avand la baza V8 engine, Node.js aduce performante imbunatatite fata de un web server clasic deoarece utilizeaza concomitent atat un interpret, care are ca scop analizarea codului si identificarea anumitor sabloane de functii sau variable care se executa sau sunt declarate frecvent si le compileaza instant, fara a parcurge intregul document precum s-ar efectua intr-un mediu de programare bazat doar pe compilator. In cazul in care tipurile parametrilor din functii se modifica pe parcurs atunci compilatorul le recompileaza cu noua configuratie. Aceasta metoda a fost dezvoltata de Google sub numele de Just in Time (JIT) Compilation.

Faza de executie a codului se bazeaza pe Memoria Heap si pe Call Stack. Memoria Heap este locul unde este atribuita memoria tuturor variabilelor si functiilor. Call Stack este o stiva unde sunt introduse fiecare apel al functiilor, iar dupa executie sunt retrase. Atunci cand interpreterul parcurge codul, informatiile sunt stocate sub forma de obiecte, intr-o structura de tip Map, unde cheia este definita de catre byte code, iar valorile sunt reprezentate de functiile care urmeaza sa execute byte code-ul, astfel salvand multa memorie.

Garbage collectorul este un aspect principal al programarii si metodele de functionare a acestuia au adus imbunatatiri semnificative ale utilizarii memoriei. Cele 3 metode ale mediului Node.js de a utiliza garbage collectorul sunt:

* Colectarea paralela: Thread-ul principal de rulare se foloseste de ajutorul unor thread-uri auxiliare pentru curatare, astfel thread-ul principal este oprit pentru o perioada scurta de timp
* Colectarea incrementala: Thread-ul principal alterneaza intre curatare si rulare a codului
* Colectarea concurenta: Thread-ul principal nu este conturbat de curatarea efectuata de thread-urile ajutatoare

Dupa cum spune si creatorul Node.js, scopul principal al mediului de rulare JavaScript a fost “Creearea de aplicatii web real-time cu abilitatea de adaugare”. Node.js a fost creat ca fiind un mediu non-blocking, bazat pe evenimente de tip Input/Output.

Astfel, Node.js a adus in plus, fata de cei 20 de ani in care aplicatiile de tip stateless-web bazate pe clasicul request-response, conceptele de tip real-time, comunicare in doua sensuri, in mediul web.

Menirea Node.js nu este pentru efectuarea operatiilor costisitoare din punct de vedere al puterii de computare, ci dezvoltarea rapida a solutiilor web, asigurarea scalabilitatii, capabila sa se ocupe de un numar mare de conectari simultane cu transfer de date intens.

Conceptul de programare asincrona sta la baza buclei de evenimente a mediului de rulare Node.js, astfel incat operatiile de tip I/O nu blocheaza thread-ul principal si se pot executa simultan pe threaduri separate. Acestea sunt numite operatii asincrone, iar in rulare programul nu asteapta ca acestea sa fie finalizate ci se trece mai departe la urmatorul task.

Un alt element important al programarii utilizand Node.js este conceptul de “callback”, care de asemenea sta la baza programarii functionale. “Callback” este o functie care se transmite ca argument intr-o alta functie, urmand sa fie executata la finalul unei actiuni, la un timp ulterior.

Avand in vedere ca majoritatea sistemelor de operare au la baza un kernel multi-thread care poate executa mai multe operatii simultan, iar Node.js este bazat pe evenimente, la finalizarea fiecarui task este semnalat un eveniment, iar urmeaza rularea unui “callback” corespunzator.

Operatiile nefinalizat sunt pastrate in bucla de evenimente care asteapta semnale din partea acestora.

<https://www.freecodecamp.org/news/node-js-what-when-where-why-how-ab8424886e2/>

<https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js>

WebSockets

WebSocket-urile sunt un protocol de comunicare bazat pe canale de comunicare de tip full-dupex, care pot interschimba informatia pintr-o singura conexiune de tip TCP. Acestea au aparut in 2008 ca un suport pentru aplicatiile web de tip real-time, inainte neavand un suport bine definit prin intermediul protocolului clasic HTTP bazat pe request-response.

Inainte de WebSockets principala modalitate de transmitere continua a datelor se realiza prin intermediul unei proceduri denumite “long pooling”, care consta in deschiderea unei conexiuni de tip HTTP cu serverul si metinerea conexiunii cat timp aceasta era necesara. Dezavantajele acestei metode sunt:

* Serverul nu stie cand clientul realizeaza o cerere a datelor, astfel ca acesta poate trimite date la momente nenecesare de tip, doar pentru a mentine deschisa conexiunea de tip HTTP
* Acest lucru se poate intampla de asemenea din partea clientului, astfel se dezvolta o supraincarcare a canalului de comunicatie care nu este necesara, iar datele pot deveni inconsistente

Se poate considera ca WebSocket-urile au introdus o noua gama de posibile aplicatii prin care conexiunea dintre client si server poate fi mentinuta permanent.

Securitatea protocolului consta in autentificarea de tip handshake care deschide canalul de comunicatie in care se trimit mesaje suprapuse peste conexiunea de tip TCP. Inainte de initializarea propriu zisa a unui WebSocket este necesara creerea unei conexiuni de tip HTTP care printr-o cerere trimisa de client catre server, avand un parametru special de tip “Header” care contina o cheie unica, criptata utilizand SHA-1 hash, avertizand serverul ca vrea sa intieze o comunicare prin WebSocket. Serverul valideaza cheia primita, daca raspunsul este unul pozitiv se realizeaza autentificarea de tip handshake, prin raspuns fiind trimisa o cheie de accepta criptata prin aceeasi modalitate, sau poate fi unul negativ.

Datele pot fi trimise de la client la server si vice versa. Acestea sunt transmise in format binar, in marimi de tip byte, printr-un buffer. Trimiterea se face la momente exacte de timp pentru a mentine consistenta datelor dintr-un capat al transmisie la celalalt.

<https://ably.com/topic/websockets>

<https://arxiv.org/pdf/1409.3367.pdf>

<https://www.html5rocks.com/en/tutorials/websockets/basics/>

Multimodal Biometrics

Termenul de biometric se refera la masurarea si analizarea unor identificatori unici umani cu scopul autentificarii intr-un sistem de securizare complex. Pe langa sistemele de securitate specializate, autentificarea bazata pe date biometrice devine din ce in ce mai comuna in viata omului, aducand o simplitate in utilizarea aplicatiilor/sistemelor nefiind necesara memorarea unei parole sau alte date de securizare.

Procesul de autentificare bazat pe datele biometrice consta in compararea datelor colectate de la un posibil utilizator si trecerea acestora printr-un algoritm de criptare tip cale unica sub format binar si compararea ulterioara a acestora cu anumite sabloane create initial de catre utilizator.

Autentificarea se face pe baza unui scor denumit “matching score” care reprezinta procentul de acuratete al comparatiei dintre datele colectate curent si cele sablon.

Date biometrice utilizate frecvent:

* Recunoastere faciala
* Amprente
* Iris
* Voce

Dezavantajul sistemelor unimodal biometrice este acela ca autentificarea poate sa dea gres in cazul in care utilizatorul a adus modificari asupra identificatorului, de exemplu arsura la deget, purtarea de lentile de contact etc..

Pentru a evita acest caz se utilizeaza sistemele de autentificare multimodale, care utileaza doua sau mai multe date biometrice ale aceluiasi utilizator.

De cele mai multe ori sistemul multimodal face comparatia datelor biometrice independent cu sabloanele stocate in baza de date, acestea ulterior fiind fuzionate intr-un algoritm de “matching score”, iar autentificarea se face pe baza scorului rezultat utilizat de un algoritm de decizie.

…

10.1109/ICOEI.2019.8862563

<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.6907&rep=rep1&type=pdf>

10.1109/ICECDS.2017.8389858

10.1109/ICRTIT.2014.6996159

10.1109/ISCO.2017.7856044

<https://www.sciencedirect.com/topics/computer-science/multimodal-biometric>

<https://www.researchgate.net/publication/254029837_Multimodal_Biometrics_--_Sources_Architecture_and_Fusion_Techniques_An_Overview>

https://www.techtarget.com/searchsecurity/definition/biometrics