4. Data Access

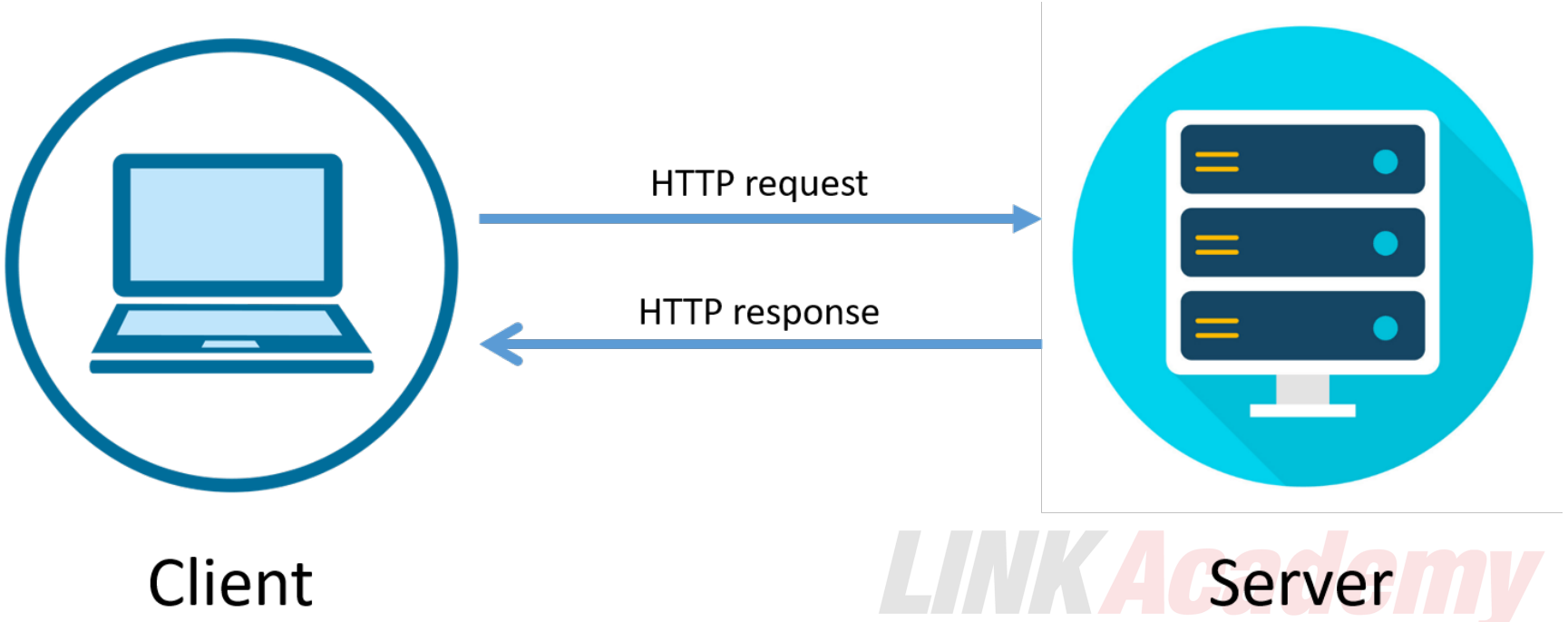
Adrian Adiaconitei

LINK Academy

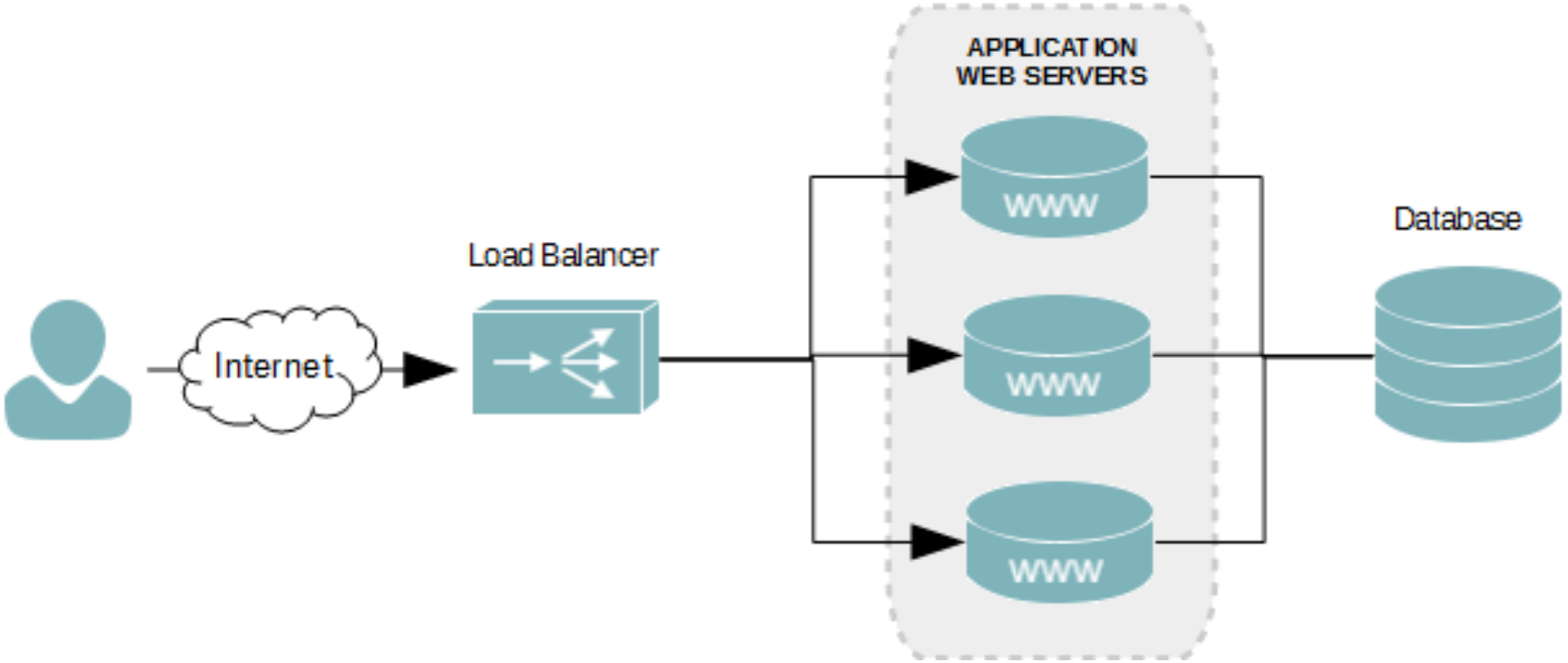
Obiective

- ✓ HTTP Request/Response model
- ✓ WebSocket
- ✓ Cookie-uri / Sesiuni
- ✓ Local storage
- ✓ Session storage
- ✓ IndexedDB

HTTP Request/Response model

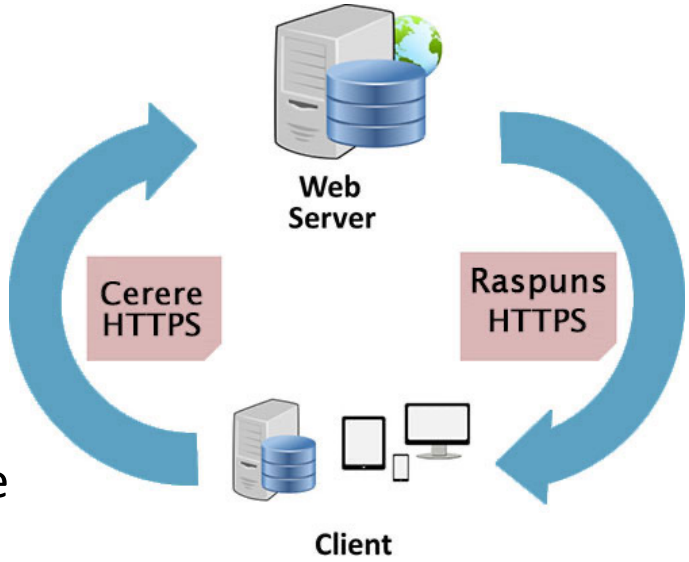


HTTP Request/Response model



Protocolul HTTP/HTTPS

- ✓ **HTTP**: Este un set de reguli de comunicare intre client (browser) si server. Comunicarea **HTTPS** este criptata
- ✓ **Adresa IP** – mod de recunoastere a serverului
- ✓ **Port -** numar de la 0 - 65535
  - Determina modul de comunicare
  - Se leaga cu adresa IP cu care formeaza o adre de retea completa



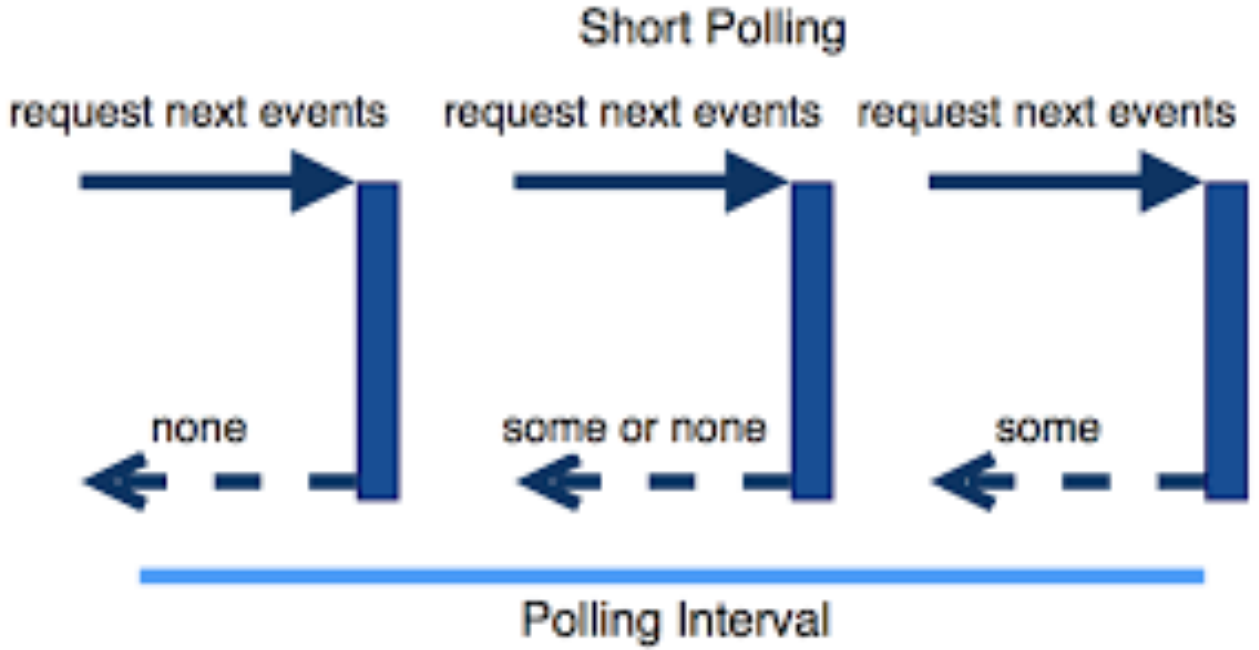
- ✓ **TCP(Transmission Control Protocol)** asigura livrarea ordonata a unui flux de octeți de la un program de pe un computer la alt program de pe un alt computer. **TCP**: nivel scăzut, bidirecțional, full-duplex și nivel de transport garantat. Fără suport pentru browser

HTTP Request/Response model

- ✓ **Short polling:** este un temporizator bazat pe cereri AJAX care apelează la întârzieri fixe.
- ✓ Clientul face o cerere către server
- ✓ Serverul poate răspunde în două moduri:
  - ✓ Trimite un răspuns gol
  - ✓ Trimite obiectul de date în corpul său (Obiect JSON)
- ✓ De îndată ce un client primește răspunsul de la server, acesta va aștepta câteva secunde și va repeta procesul de mai sus.
- ✓ Dezavantaje: Efectuarea de solicitări repetate către server irosește resurse deoarece fiecare nouă conexiune de intrare trebuie să fie stabilită, anteturile HTTP trebuie transmise, trebuie efectuată o interogare pentru date noi și trebuie generat și livrat un răspuns

HTTP Request/Response model

✓ **Short polling:**



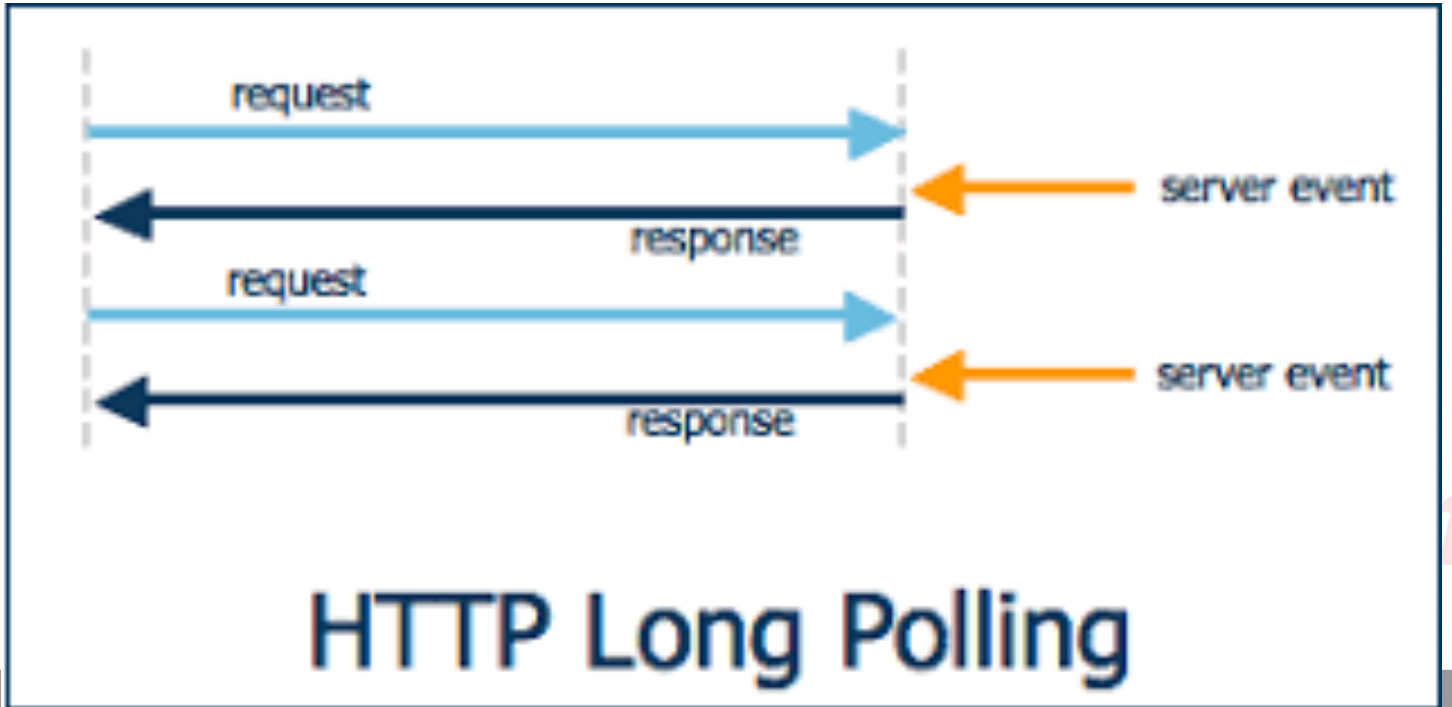
HTTP Request/Response model

- ✓ **Long Polling:** este un mecanism în care clientul solicită în mod continuu serverului informații noi folosind cereri HTTP obișnuite și serverul își blochează răspunsul atunci când nu are nimic nou de raportat.

- ✓ Clientul face o cerere către server
- ✓ Serverul poate răspunde în două moduri:
  - ✓ Dacă are câteva date noi disponibile, poate răspunde imediat.
  - ✓ Dacă nu are date noi, va menține acea conexiune deschisă pentru o perioadă de timp și atunci când primește date noi, va răspunde cu date actualizate.

HTTP Request/Response model

✓ **Long Polling:**



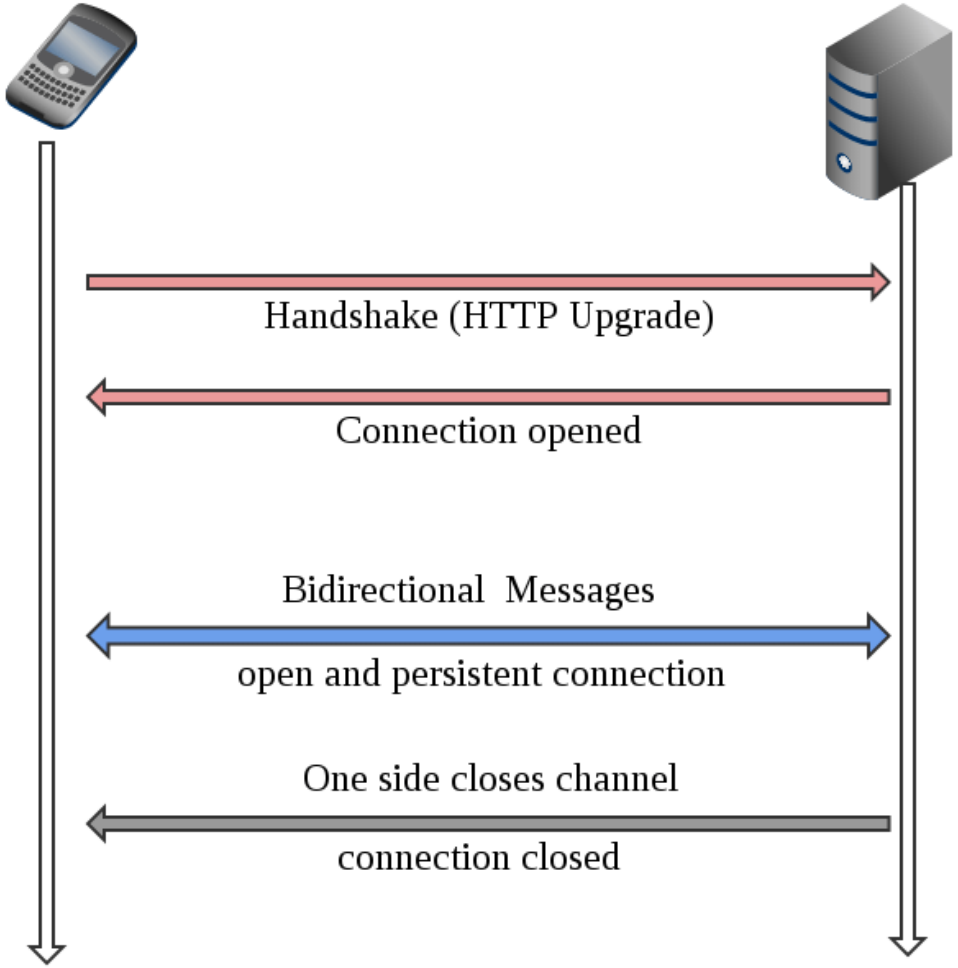
WebSocket

- ✓ **WebSocket** este un protocol de comunicație pentru computer care oferă canale de comunicație full-duplex (bidirectionale) printr-o singură conexiune TCP.
- ✓ Protocolul WebSocket permite interacțiunea între un client și un server web cu cheltuieli generale mai mici, oferind transfer de date în timp real de la și către server. WebSockets menține conexiunea deschisă, permițând transmiterea mesajelor înainte și înapoi între client și server. În acest fel, poate avea loc o conversație continuă între client și server.

LINKAcademy

WebSocket

(*ws://* si *wss://*) vs **http** si **https**.



WebSocket

Aplicatie:
server.js
client.js
npm install ws
node server.js
node client.js

WebSocket

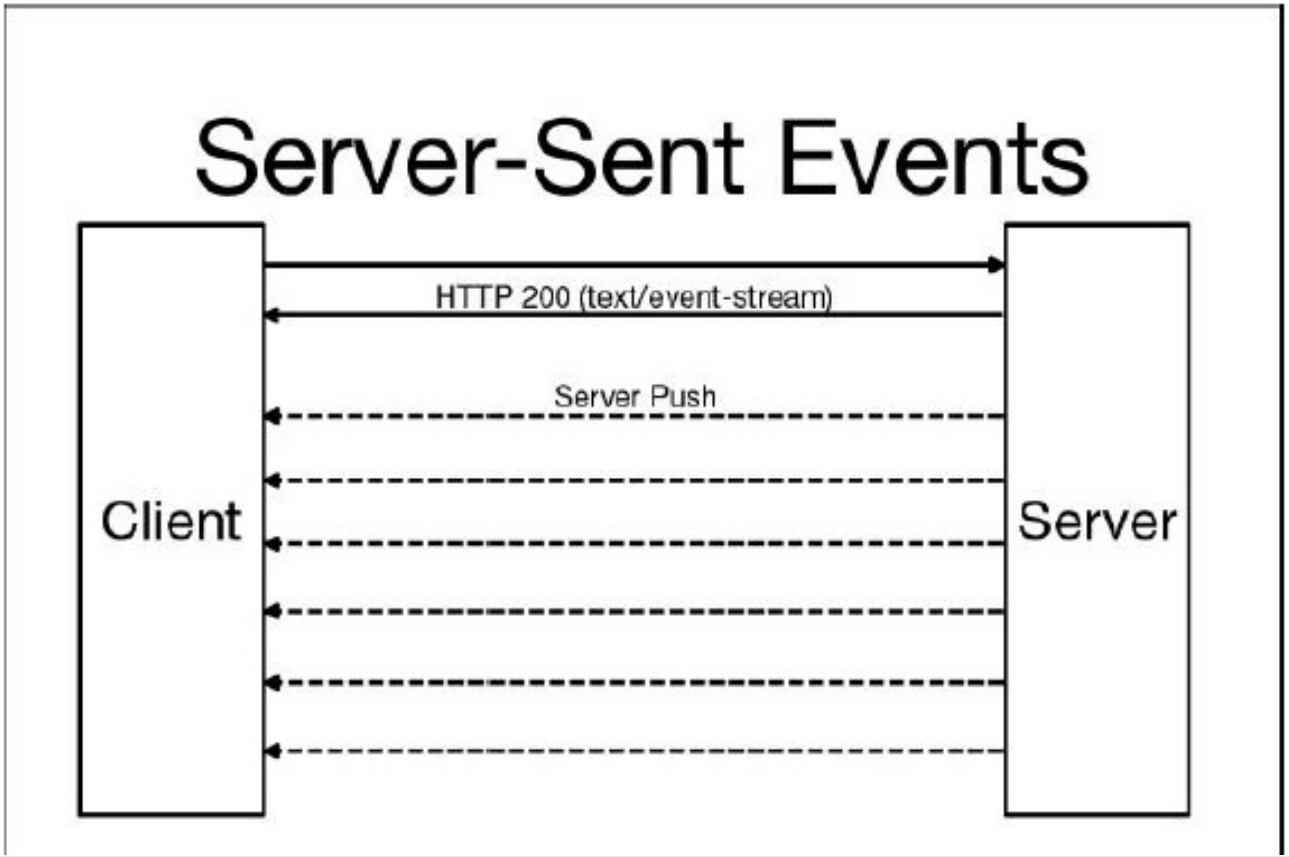
Aplicatie:
node.js,
Express,
Websockets
MySQL

Mychat:
server.js

HTTP streaming

- ✓ Streaming HTTP: o varietate de tehnici (multipart/chunked response) care permit serverului să trimită mai mult de un răspuns la o singură cerere client. W3C standardizează acest lucru ca evenimente trimise de server folosind un tip MIME text/event-stream.
- ✓ API-ul browserului (care este destul de similar cu API-ul WebSocket) se numește API-ul EventSource.
- ✓ Poate fi o alternativă la WebSockets. Problema principală din perspectivă în timp real este că un intermediar poate întrerupe conexiunea – fie prin timeout, fie pur și simplu pentru că deservește cereri multiple

HTTP streaming



HTTP streaming

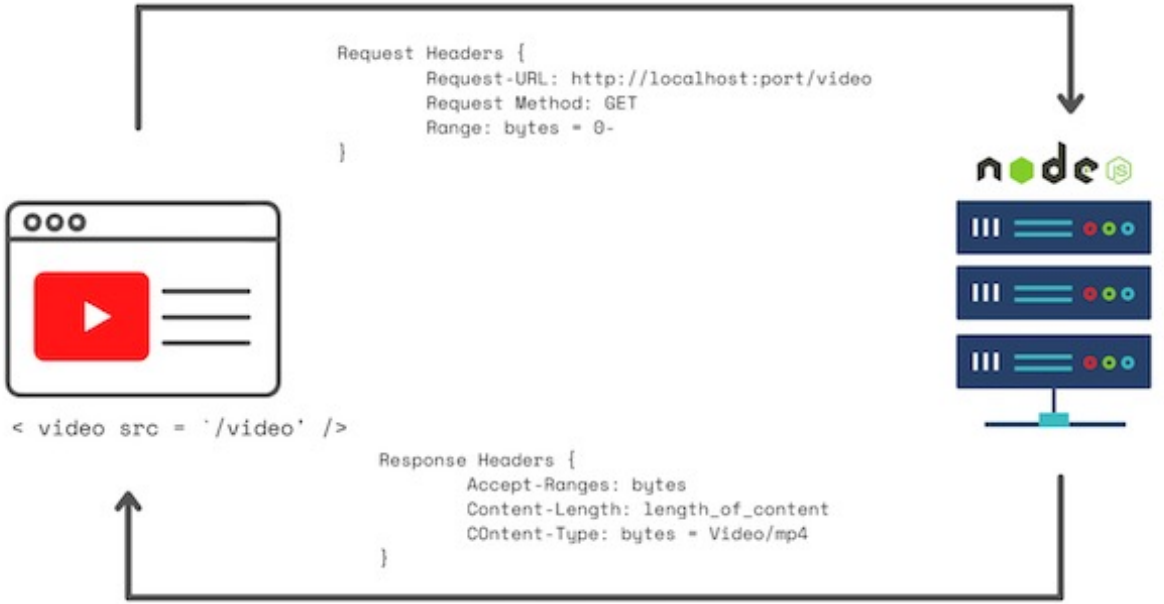
Aplicatie:

git clone https://github.com/thesmartcoder7/video_streaming_server.git .

npm install

npm start

http://localhost:8000



Cookies:

Un cookie este o variabilă păstrată în calculatorul vizitatorului.
Cu JavaScript, puteți crea și extrage cookies.

Exemple de cookies:

1. **Numele utilizatorului** – Prima dată când utilizatorul va vizitează pagina trebuie să-și completeze numele. Numele este stocat într-un cookie. Numele este recuperat dintr-un cookie.

2. Parola –Parola este memorată într-un cookie. Data viitoare când vizitatorul ajunge în pagina, parola poate fi recuperata dintr-un cookie.

3. Data calendaristică – Prima dată când utilizatorul va viziteaza pagina, data curenta este memorată într-un cookie

Cookies:

Crearea și memorarea unui cookie

-memorează numele vizitatorului, apoi vom folosi numele memorat în variabilă cookie pentru a afisa un mesaj de bun venit. Prima dată vom crea o funcție care memorează numele vizitatorului într-o variabilă cookie:

```
function setCookie(c_name,value,expiredays)    {
var exdate=new Date();
exdate.setDate(exdate.getDate()+expiredays);
document.cookie=c_name+ "=" +escape(value)+ ((expiredays==null) ? "" :
";expires="+exdate.toGMTString());
}
```

Parametrii funcției reprezinta numele și valoarea cookie-ului și numărul de zile până când acesta expira. Datele nume, cookie-ului şi data expirării sunt memorate într-un obiect **document.cookie**.

Cookies:

```
function getCookie(c_name)    {
        if (document.cookie.length>0)    {
                c_start=document.cookie.indexOf(c_name + "=");
                if (c_start!=-1)    {
                        c_start=c_start + c_name.length+1;
                        c_end=document.cookie.indexOf(";",c_start);
                        if (c_end==-1) c_end=document.cookie.length;
                        return
                        unescape(document.cookie.substring(c_start,c_end));
                }
        }
        return "";
}// functia verifica daca un cookie este setat
```

Cookies:

Funcția care afiseaza un mesaj de bun venit dacă cookie-ul este setat și o casetă prompt care cere numele vizitatorului, în caz contrar:

```
function checkCookie()    {
        username=getCookie('username');
        if (username!=null && username!=""){alert('Welcome again  '+username+'!');
        }
                else   {
                        username=prompt('Please enter your name:',"");
                        if (username!=null && username!="")   {
                                setCookie('username',username,365);
                        }
                }
}// https://github.com/aadiaconitei/websitejs/blob/main/curs2/aplicatia3.html
```

Cookies cu Node.js:

Modul: cookie-parser

```
app.get('/setcookie', (req, res) => {
      res.cookie(`Cookie token name`,`encrypted cookie string Value`,{
      maxAge: 5000,// expires works the same as the maxAge
      expires: new Date('2022-12-01'),
     secure: true,
     httpOnly: true,
     sameSite: 'lax' });
     res.send('Cookie have been saved successfully');
 });
```

Cookies cu Node.js:

Modul: cookie-parser

```
// get the cookie incoming request
app.get('/getcookie', (req, res) => {
        //show the saved cookies
        console.log(req.cookies)
        res.send(req.cookies);
});
// delete the saved cookie
app.get('/deletecookie', (req, res) => {
        //show the saved cookies
        res.clearCookie() res.send('Cookie has been deleted successfully’);
 });
```

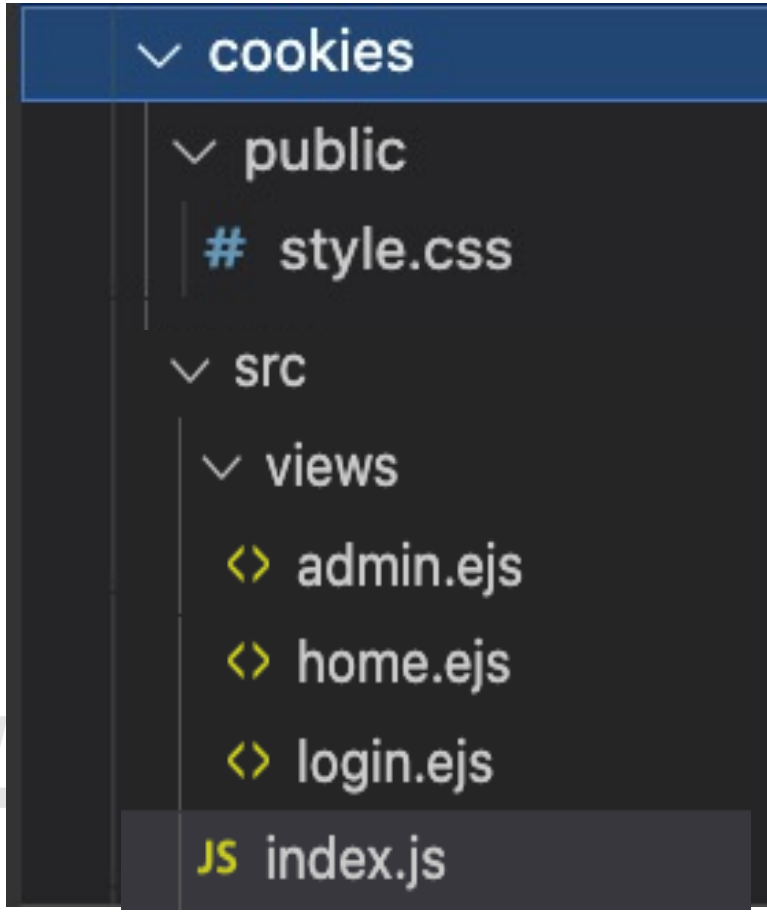
Cookies cu Node.js:

Aplicatie: Login cu cookies

mkdir cookies
npm init –y
npm i --save cookie-parser express ejs helmet nodemon

```
"scripts": {
        "dev": "nodemon src/index.js",
        "start": "node src/index.js"
}
```

index.js
npm run dev
http://localhost:4000/



Session **cu** Node.js**:**

- ✓ Un site web se bazează pe protocolul HTTP
- ✓ HTTP este un protocol **stateless** ceea ce înseamnă că la sfârșitul fiecărui ciclu de cerere și răspuns, clientul și serverul uită unul de celălalt.

Aici intervine sesiunea.

- ✓ O sesiune va conține câteva date unice despre acel client pentru a permite serverului să îl țină mine.

Session **cu** Node.js:

✓ Cum funcționează sesiunile

Când clientul face o cerere de conectare la server, serverul va crea o sesiune și o va stoca pe **server**.

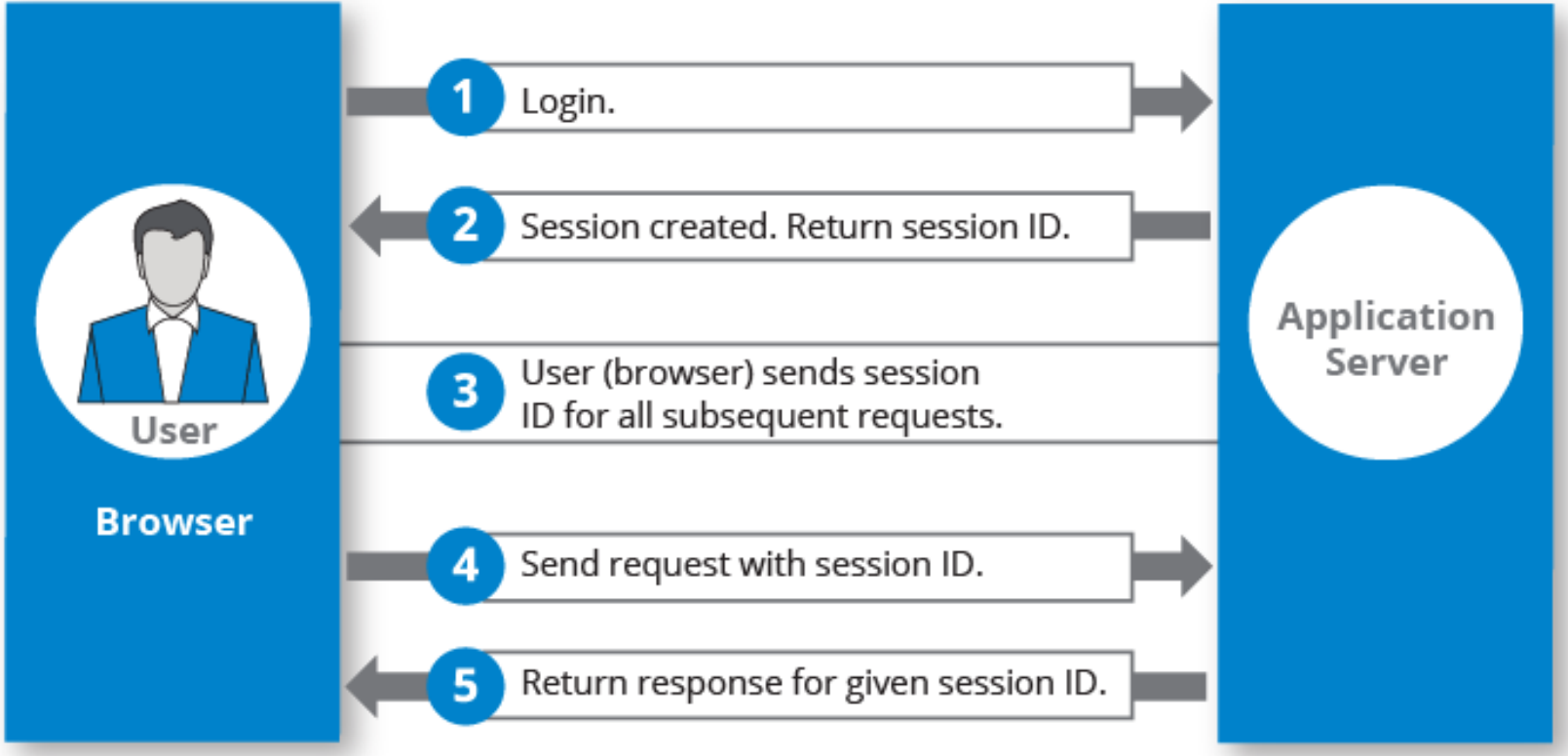
Când serverul răspunde clientului, acesta trimite un cookie. Acest cookie va conține ID-ul unic al sesiunii stocat pe server.

Acest cookie va fi trimis la fiecare solicitare către server.

Folosim acest ID de sesiune și căutăm sesiunea salvată în baza de date sau fișier pentru a menține o potrivire unu-la-unu între o sesiune și un cookie.

Acest lucru va face ca conexiunile cu protocolul HTTP să aibă stare.

Session **cu** Node.js:



LocalStorage:

- ✓ **LocalStorage** și **SessionStorage** sunt aproape identice, au același API.
- ✓ Diferența este că, cu **SessionStorage**, datele sunt persistente doar până când fereastra sau fila este închisă.
- ✓ Cu **LocalStorage**, datele sunt păstrate până când utilizatorul șterge manual memoria cache a browserului sau până când aplicația dvs. web șterge datele.
- ✓ **LocalStorage** și **SessionStorage** au aceeași sintaxă.

LocalStorage:

- ✓ **Setare**

```
let key = 'Item 1';
localStorage.setItem(key, 'Value’);
```

- ✓ **Citire**

```
let myItem = localStorage.getItem(key);
```

- ✓ **Editare**

```
localStorage.setItem(key, 'New Value’);
```

- ✓ **Ștergere**

```
localStorage.removeItem(key);
localStorage.clear();
```

LocalStorage:

- ✓ **Setare**

```
let myObj = { name: 'Skip', breed: 'Labrador' };
localStorage.setItem(key, JSON.stringify(myObj));
```

- ✓ **Citire**

```
let item = JSON.parse(localStorage.getItem(key));
```

Verificam daca este suportat în browser:

```
if (window.localStorage) {
    // localStorage supported
}
```

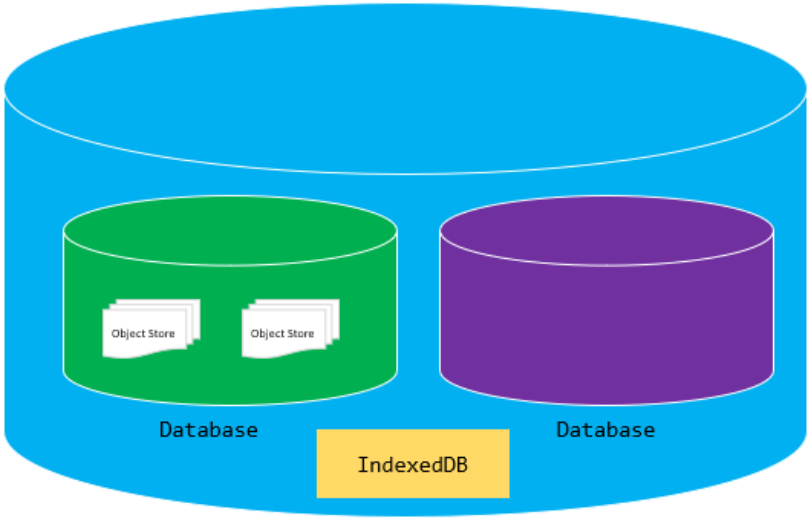
https://github.com/aadiaconitei/websitejs/tree/96aa505156a7ae83a3748c701ce77d8f52928d64/curs3/ToDoCumparaturi

https://caniuse.com/namevalue-storage

IndexedDB:

- ✓ IndexedDB vă permite să stocați în mod persistent datele folosind perechi cheie-valoare.
- ✓ Valorile pot fi de orice tip JavaScript, inclusiv boolean, număr, șir, nedefinit, nul, dată, obiect, matrice, regex, blob și fișiere.
- ✓ IndexedDB vă permite să creați aplicații web care pot funcționa atât online, cât și offline.
- ✓ Este util pentru aplicațiile care stochează o cantitate mare de date și nu au nevoie de o conexiune la internet persistentă.
- ✓ Spre deosebire de localStorage și sessionStorage, valorile stocate în IndexedDB pot fi structuri complexe precum obiecte și fișiere.

IndexedDB:



Database

objectStore

key1: value1
key2: value2
key3: value3
key4: value4
key5: value5

objectStore

key1: value1
key2: value2
key3: value3
key4: value4
key5: value5

objectStore

key1: value1
key2: value2
key3: value3
key4: value4
key5: value5

IndexedDB:

✓ **IndexedDB este tranzacțional**

Fiecare citire și scriere în bazele de date IndexedDB are loc întotdeauna într-o tranzacție.

✓ **API-ul IndexedDB este în mare parte asincron**

Operațiunile IndexedDB sunt asincrone. Folosește evenimente DOM pentru a vă anunța când se termină o operațiune și rezultatul este disponibil.

✓ **IndexedDB este un sistem NoSQL**

Folosește interogarea care returnează un cursor. Apoi, puteți folosi cursorul pentru a repeta setul de rezultate.

IndexedDB:

```
let openRequest = indexedDB.open(name, version);
let openRequest = indexedDB.open(”mydb", 2);
openRequest.onupgradeneeded = function() {
        let db = openRequest.result;
        switch(event.oldVersion) { // existing db version
                case 0: // version 0 means that the client had no database
                case 1: // client had version 1 // update
        }
};
openRequest.onerror = function() {
        console.error("Error", openRequest.error);
};
openRequest.onsuccess = function() {
let db = openRequest.result; // continue working with database using db object
};
```

IndexedDB:

```
openRequest.onsuccess = function() {
        let db = openRequest.result;
        db.onversionchange = function() {
                db.close();
                alert("Database is outdated, please reload the page.")
        };
        // baza de date este ok și poate fi folosită
};
openRequest.onblocked = function() {
// acest eveniment nu ar trebui să se declanșeze dacă gestionăm corect
//onversionchange înseamnă că există o altă conexiune deschisă la aceeași bază
//de date și nu a fost închis după declanșarea db.onversionchange pentru el
};
```

IndexedDB:

Aplicatie:

Git clone https://github.com/theShagod/indexeddb-mysql-sync-testing.git .

Npm install

Editam config.json

Node server.js

http://localhost:5050/

Resurse:

https://dev.to/alexeagleson/how-to-use-indexeddb-to-store-data-for-your-web-application-in-the-browser-1o90

https://javascript.info/indexeddb#open-database