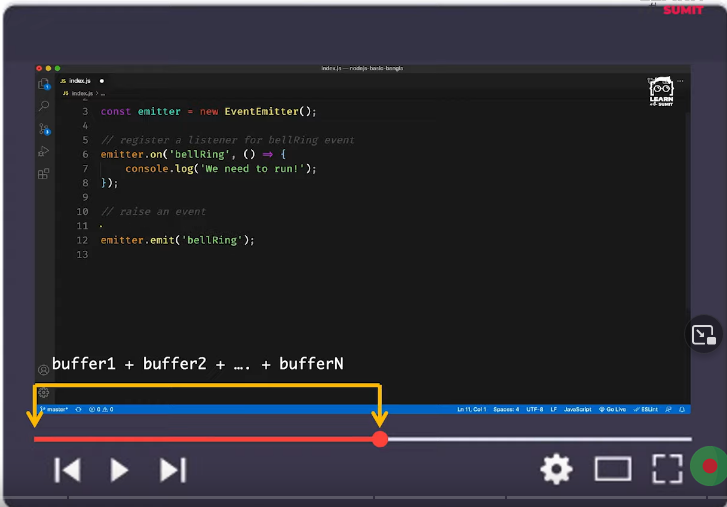
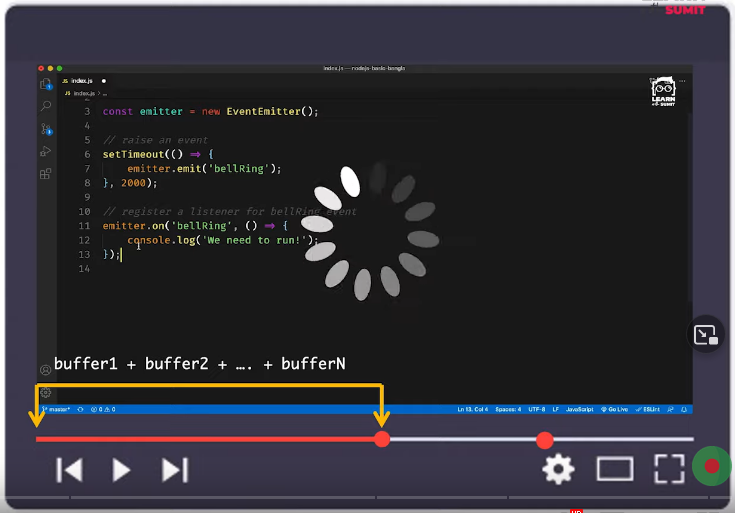
**Stream and Buffer**

Stream and buffer amader porichito cz video streaming platformgula e.g. youtube, Netflix etc e amra visit kortesi. Youtube e ekhon je video ta dekhtesi sheta real time e dekhtesi. Stream kore dekhtesi.

Video ba text hok shobgulai kintu data. Data ta stream er moto kore ashtese. Jokhon vdo er play button e chap dei tokhoni streaming ta shuru hoy. Stream means flow or probaho. Orthat data ta ektu ektu kore probahito hoye amader kache ashtese. Purata ek bare chole ahstese na. playbar e jototuku red dekhtesi toto tuku load hoye gese.

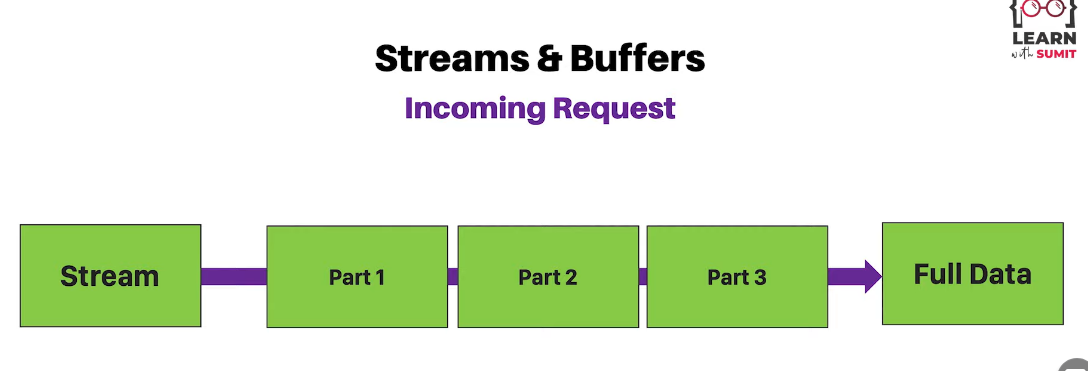


Ei je data ta esheche tar ei ashar process ta ke bole stream. Ei je joto tuku ami load kore dekhe felsi shegula choto choto kichu buffer er sum. Ei buffer ke ekta data type hishebe dhora jay like string, number. Ektu ektu kore buffer amader kache ashte thake. Shei buffergula ultimately client end (browser) er ashar por amader kache display hocche. Chunk e chunk e vag hoye data amader kache ashche. Ekhon vdo ta ke ektu tene shamne niye jabo. Tokhon loading sign dekhay. Ei loading ta ke amra normally boli buffering hoitese.



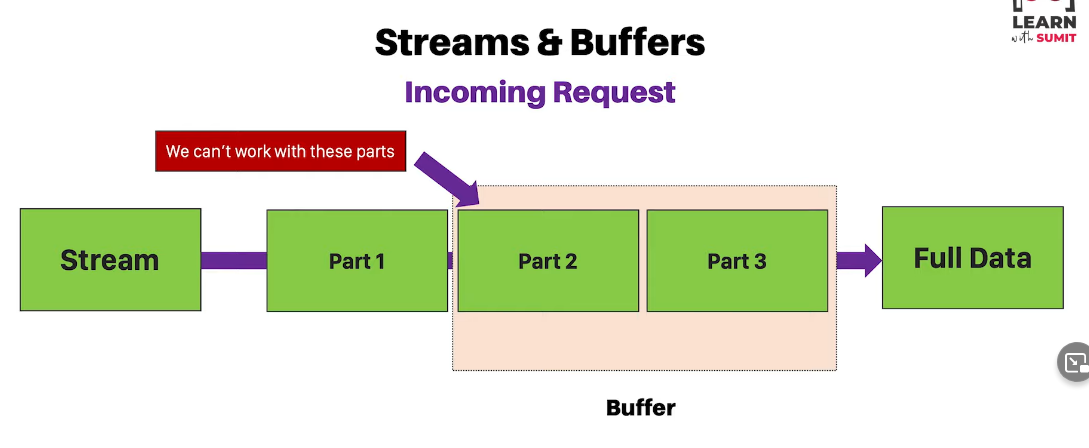
Ami jokhon vdo ta hothat taan dilam amar secuence of stream interrupted hoye ami taan diye je jaygay niye gelam oi jaygatar proyojoniyo buffer er jonno wait kortese.

Ekhon stream ebong buffer er technical explanation dekhbo. Amader data ta ashe kono na kono server theke. For e.g. youtube er video data ta youtube er server theke amader kache ashtese. Eta ke amra boli incoming request. Orthat youtube theke ekta incoming request ashche amar browser e.



Left side e je stream dekhtesi sheta dhorlam youtube er data ta. Finally amar browser full data ta ashtese. Pura video ta part e part e vag hoye ashtese.

Ekhon youtube er kotha chinta na kore jodi ekta node application er kotha chinta kori tahole shekhaneo beparta same. Suppose, client amar node server e ekta 10MB size text file read korar jonno request korse. Jodi ami streaming kore na kori tahole client ke request korar por amar server theke pura data ta client end i.e. tar browser e jete jototuku time lage toto tuku shomoy boshe thakte hobe. Orthat pura data load howar shomoy tuku take wait kore thakte hobe. Ei jinish ta ke performant way te korar jonnoi streaming of data concept ta ashce. Amra part e part e vag kore data pathabo server theke. Amra part1, part2, part3 ja dekhtesi shegula broken parts. Egula binaryr moto kore ashe. Egula niye amra kaj korte pari na. egula binary data.

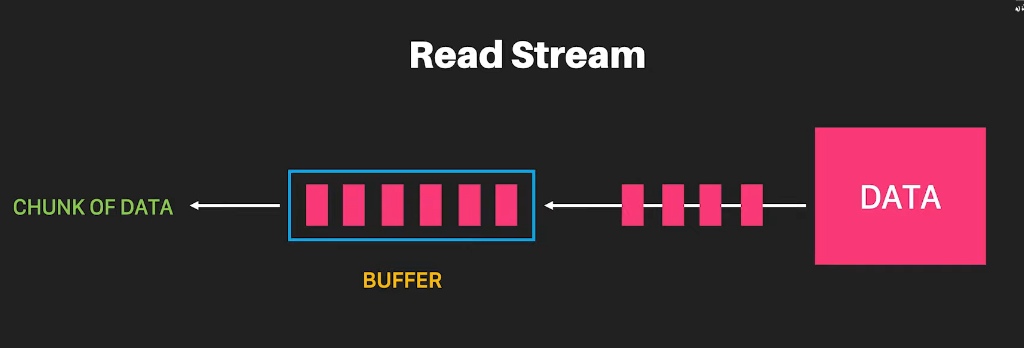


Ei partgular moddhe kono duita part ba tinta part mile ekta buffer ba encapsulated packet er moto toiri hoy. Ekadhik part mile ekta buffer ba packet toiri hoy. Tarpor ei buffer ta basically client end e chole jay. Amra ei buffer gula ke niye kaj korbo. Ei buffer ke bole Buffer datatype. Eta ekta object. Eta niyei amra kaj korbo.

So ei buffer er duita use case hote pare.

1. Jokhon amra data read kortesi. Sheta ke bole read stream. Jemon youtube er video dekha.
2. Write stream. Mane ami data paisi. Ba ami kono boro file system theke file read kore felsi. Ekhon ami abar kono ekta file e write korbo. Ekhon 10MB size er file write korte geleo buffering kore kore korte hobe.

First e read stream er explanation ta abar dekhbo.



Amader data ta streaming hoye ashe ektu ektu kore. Choto choto pink partgulo egula ekekta part of data. Koyekta part mile ekta buffer create kortese. Koyekta part mile jokhoni ekta buffer complete korbe tokhoni sheta client end e chole ashbe ekta chunk of data er moto. Same jinish ta repeat hote thakbe jotokkhon porjonto na complete data ta client er kache chole ashe.

Ekhon sheta code er maddhome dekhbo.

Nodejs Practice folder e bigdata.txt name ekta file niye tate onek text data nilam. Amar purpose ei file ta ke streaming way te read kora. Ind.js file e fs module ta require korlam. Er age amra dekhsilam fs.readFile() diye async way te data read korte pari. Shekhetre amra je data ta paisilam sheta streaming way te chilo na. kintu ekhon streaming use kore data read korbo.

Ind.js …

*const* fs = *require*("fs");

*const* ourReadStream = fs.*createReadStream*(`${\_\_dirname}/bigdata.txt`);

ourReadStream.*on*("data", (*chunk*) *=>* {

  console.*log*(chunk);

});

Ekhane createReadStream() use kore ekta stream object toiri kortesi. Etar moddhe first e she file path ney. mane kon file ta she read korbe. orthat bigdata.txt file tar jonno ekta read stream toiri hoye gelo. Ei read stream create howar por she amader ektu ektu kore buffer dibe ar event dite thakbe. ourReadStream.on(‘data’) --- ekhane amra ‘data’ event tar jonno event listen kortesi. ourReadStream amader continuously data diye jaitese. Ekta kore buffer ashte thakbe ar ekta kore ‘data’ event listen korbo.

ourReadStream.on(‘data’, (chunk)=>{

console.log(chunk)})

Ekhane listener er callback function er moddhe amra buffer ba chunk gula ke dekhar besbostha korlam. Node ind korle dekha jaitese chunk e chunk e vag hoye data ashtese. Prottekta chunk ekekta buffer. Ei buffer data ba binary data niye amra kaj korte parbo na. createReadStream er upor cursor nile dekha jabe BufferEncoding name ekta jinish ache. So ei parameter hishebe ‘utf8’ pass kore dile console ekdom data tai pabo.

Ind.js …

*const* fs = *require*("fs");

*const* ourReadStream = fs.*createReadStream*(`${\_\_dirname}/bigdata.txt`, "utf-8");

ourReadStream.*on*("data", (*data*) *=>* {

  console.*log*(data);

});

Abar eto kichu na kore chunk.toString() kore dileo text paoa jay.

Ind.js …

*const* fs = *require*("fs");

*const* ourReadStream = fs.*createReadStream*(`${\_\_dirname}/bigdata.txt`);

ourReadStream.*on*("data", (*chunk*) *=>* {

  console.*log*(chunk.*toString*());

});

Ebar niche concole.log(“Hello Guys”) likhe abar o/p dekhlam.

*const* fs = *require*("fs");

*const* ourReadStream = fs.*createReadStream*(`${\_\_dirname}/bigdata.txt`);

ourReadStream.*on*("data", (*chunk*) *=>* {

  console.*log*(chunk);

});

console.*log*("Hello Guys");

ekhane “Hello Guys” first e print hoise. Ekhane ourReadStream event ta fire hobe jokhon data ta tar kache received hoy. Nodejs code porte porte jokhon ourReadStream.on() pay tokhoni kintu executed hoy na. eta ekta just listener. She just listen korar jonno boshe ache. Execute hocche “Hello Guys” age. Bigdata.txt read kore first chunk ta jokhon received hoy tokhon ekbar callback function er console.log() print hoy. 2nd chunk paile arekbar console.log() hoy. Etai nodejs er default system.

Ekhon amra http module diye ekta server create kore dekhabo amader server e jokhon ekta incoming request ashe shei request keo amra chunk e chunk e receive korte pari. Et bujhar jonno arekta example ekhon dekhbo.

Amader server er kache jokhon request ashe tokhon sheta je by default stream akare ashe shei jinish dekhatai amar purpose.

Ind.js …

*const* http = *require*("http");

*const* server = http.*createServer*((*req*, *res*) *=>* {

  if (req.url *===* "/") {

    res.*write*("Hello Programmers");

    res.*write*("How are you?");

    res.*end*();

  } else if (req.url *===* "/about") {

    res.*write*("This is about page");

    res.*end*();

  } else {

    res.*write*("Not Found");

    res.*end*();

  }

});

server.*listen*(3000);

console.*log*("Listening on port 3000");

*//const fs = require("fs");*

*// const ourReadStream = fs.createReadStream(`${\_\_dirname}/bigdata.txt`);*

*// ourReadStream.on("data", (chunk) => {*

*//   console.log(chunk);*

*// });*

*// console.log("Hello Guys");*

Ekhane amader duita route create kora ache. Ektu age bigdata.txt file system theke read korsilam. Ekhon amar server ei ekta url rakhbo. Shei url theke oi url ta post korte parbe client. Mane ekta form field thakbe. Shekhane post kore dibe keu ekjon. Ebong ami arekta route diye shei post er post back ta receive korbo. Receive korar shomoy dekhbo bigdata.txt stream akare ahstese. Amar normal url “/” ekhane form ta rakbo.

Ind.js …

*const* http = *require*("http");

*const* server = http.*createServer*((*req*, *res*) *=>* {

  if (req.url *===* "/") {

    res.*write*("<html><head><title>Form</title></head>");

    res.*write*(

      `<body><form method="post" action="/process"> <input name="message" /></form></body> </html>`

    );

    res.*end*();

  } else if (req.url *===* "/process" && req.method *===* "POST") {

    res.*write*("This is about page");

    res.*end*();

  } else {

    res.*write*("Not Found");

    res.*end*();

  }

});

server.*listen*(3000);

console.*log*("Listening on port 3000");

*//const fs = require("fs");*

*// const ourReadStream = fs.createReadStream(`${\_\_dirname}/bigdata.txt`);*

*// ourReadStream.on("data", (chunk) => {*

*//   console.log(chunk);*

*// });*

*// console.log("Hello Guys");*

Node ind kore browser e localhost:3000 e giye input field kichu likhe enter chepe post korlei “/process” url e chole jay. Jehetu “POST” method e data ta ashar kotha shehetu req.method === “POST” kore shetao check kore nilam. “/process” e ami amar requested data ta nibo. Oneke vabte pare req.data te data ta thakte pare. Kintu sheta nai. Req.data te data petam jodi sheta ekbare chole ashto. But data ta ashtese streaming mechanism er maddhome. Tahole data ami nibo ki kore? Jehetu event er maddhome ektu ektu kore data dibe shehetu req.on() listener er maddhome data nite hobe. Listener er callback function er moddhe data dekhar bebostha korte hobe.

Ind.js …

*const* http = *require*("http");

*const* server = http.*createServer*((*req*, *res*) *=>* {

  if (req.url *===* "/") {

    res.*write*("<html><head><title>Form</title></head>");

    res.*write*(

      `<body><form method="post" action="/process"> <input name="message" /></form></body> </html>`

    );

    res.*end*();

  } else if (req.url *===* "/process" && req.method *===* "POST") {

    req.*on*("data", (*chunk*) *=>* {

      console.*log*(chunk.*toString*());

    });

    res.*write*("Thank u for submitting");

    res.*end*();

  } else {

    res.*write*("Not Found");

    res.*end*();

  }

});

server.*listen*(3000);

console.*log*("Listening on port 3000");

*//const fs = require("fs");*

*// const ourReadStream = fs.createReadStream(`${\_\_dirname}/bigdata.txt`);*

*// ourReadStream.on("data", (chunk) => {*

*//   console.log(chunk);*

*// });*

*// console.log("Hello Guys");*

Ekhon node ind kore localhost:3000 e giye input field e bigdata.txt er text ta paste kore enter chaple browser e “Thank u for submitting” ashtese ebong terminal e encoded data paitesi.

Amra kokhon bujhte parbo pura data ta stream hoye geche? Amra ‘data’ event ta listen kortesilam ekta ekta kore chunk ba buffer paoar shomoy. Pura data chole ashar beparta amader listen korte hobe ‘end’ event diye.

*const* http = *require*("http");

*const* server = http.*createServer*((*req*, *res*) *=>* {

  if (req.url *===* "/") {

    res.*write*("<html><head><title>Form</title></head>");

    res.*write*(

      `<body><form method="post" action="/process"> <input name="message" /></form></body> </html>`

    );

    res.*end*();

  } else if (req.url *===* "/process" && req.method *===* "POST") {

*const* body = [];

    req.*on*("data", (*chunk*) *=>* {

      body.*push*(chunk);

    });

    req.*on*("end", () *=>* {

      console.*log*("Stream has finished.");

*const* parsedBody = Buffer.*concat*(body).*toString*();

      console.*log*(parsedBody);

    });

    res.*write*("Thank u for submitting");

    res.*end*();

  } else {

    res.*write*("Not Found");

    res.*end*();

  }

});

server.*listen*(3000);

console.*log*("Listening on port 3000");

Ekhane Buffer nodejs er default ekta object. shetar shathe ami body array ta ke concat kore dicchi. Then sheta ke toString korle main data ta pabo.

So ami ‘data’ event listen kortesi. Je chunk ba buffer gula ashtese shegula ke ekta body array er moddhe vortesi. Finally stream shesh hoye gele ‘end’ event ta fire hobe ebong tar callback function ta run hobe. Shekhane body array theke parsedBody ke process kore antesi. So ekhon node ind kore browser localhost:3000 e giye text paste kore enter chaple thikmoto o/p paitesi.

Ekhon amra write stream ta dekhe felbo. File system e ektu age je read korar system ta dekhsilam shei vabei ekhon write kore dekhbo. ekta file read kore arekta file e wtite kore dekhabo streaming way te.

Ind.js …

*const* fs = *require*("fs");

*const* ourReadStream = fs.*createReadStream*(`${\_\_dirname}/bigdata.txt`);

*const* ourWriteStream = fs.*createWriteStream*(`${\_\_dirname}/output.txt`);

ourReadStream.*on*("data", (*chunk*) *=>* {

  ourWriteStream.*write*(chunk);

});

Ekhane ourReadStream er maddhome bigdata.txt file ta ke read korlam. Read korar por streaming way te arekta file e write kore dekhabo.

ourWriteStream namok ekta variable er moddhe createWriteStream ke niye notun ekta file ‘output.txt’ te write korar bebostha korlam.

ourReadStream.on listener er maddhome ‘data’ event listen kore ektu ektu chunk e chunke data paitesi. Ekhon ei listener er callback function er moddhe ektu ektu kore chunke chunke write korar bebostha korlam.

Ekhon node ind korle output.txt file creat holo. Tar moddhe bigdata.txt er content chole ashche.

Ei read stream ebong write stream er kaj ta aro shohoj vabe korbo pipe er maddhome. ourReadStream object ta pipe namok ekta method ache. Ei method er parameter hishebe jodi ourWriteStream ke diye dei taholeo same kajtai hobe. Orthat output.txt name ekta file create hobe. Ebong shetar content hobe bigdata.txt er content.

Ind.js …

*const* fs = *require*("fs");

*const* ourReadStream = fs.*createReadStream*(`${\_\_dirname}/bigdata.txt`);

*const* ourWriteStream = fs.*createWriteStream*(`${\_\_dirname}/output.txt`);

ourReadStream.*pipe*(ourWriteStream);

pipe, read stream, write stream ebong req, res er maddhome amra server end e je kaj ta korechilam shobkichu milaya ekta final example dekhbo.

createServer() er callback function er moddhei ekta read stream ‘myReadStream’ create kore bigdata.txt theke data read kore ekta writable stream e data write kore dibo. Ekhane amra ‘res’ ke writable stream hishebe use kortesi. Amader createServer method er moddhe je callback function pathai ebong tar je req, res duita parameter diye dei shekhane req hocche ekta readable stream ar res hocche ekta writable stream. Res hocche ami amar client ke je response ta dibo sheta.

Ind.js …

*const* http = *require*("http");

*const* fs = *require*("fs");

*const* server = http.*createServer*((*req*, *res*) *=>* {

*const* myReadStream = fs.*createReadStream*(`${\_\_dirname}/bigdata.txt`, "utf-8");

  myReadStream.*pipe*(res);

});

server.*listen*(6540);

console.*log*("Listening on Port 6540");

ekhon node ind kore browser e localhost:6540 te gele bigdata.txt er content pabo.

Ekhane client er kache data ta streaming way te gese.

Amra server er end e req, res je duita object pacchi…ekhane req hocche ekta readable stream. Amra amder client er kach theke ba onno server theke je request ta receive korbo sheta ekta readable stream.

Abar eki vabe amra client ke je response dibo shei response tao streaming way te dite hobe. Cz client er browser keo memory efficient way te response pete hobe. Youtube e je video dekhi tokhon tara response objecter moddhe stream korte thake . ebong ami amar browser e read korte thaki.