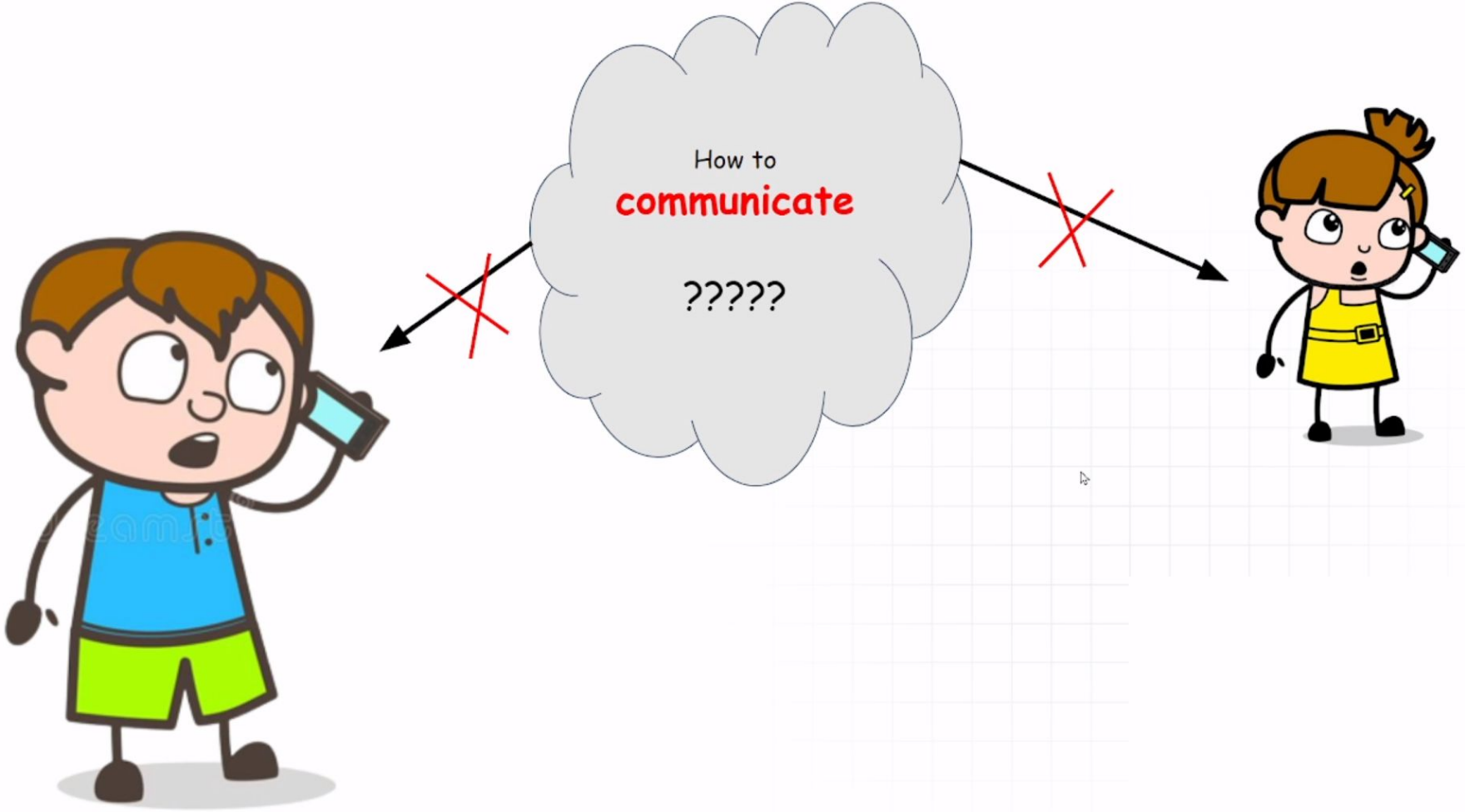
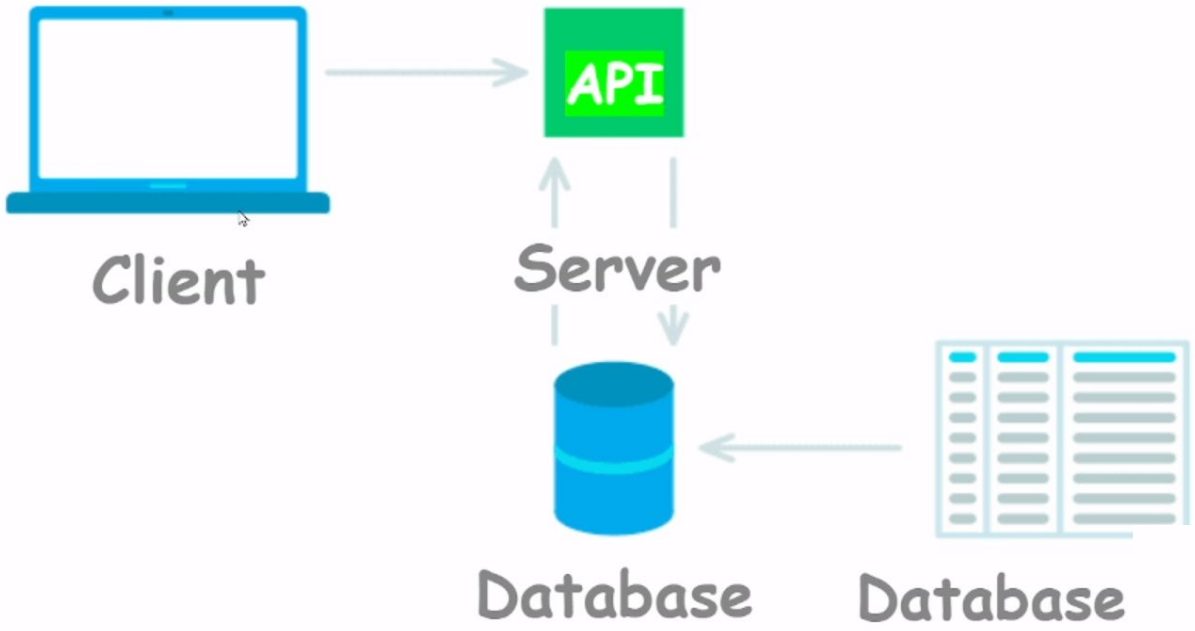
How does API connect client & server?


How to
communicate
?????

তাহলে API ছাড়া কি server & client কে connect করার কোনও way আছে?

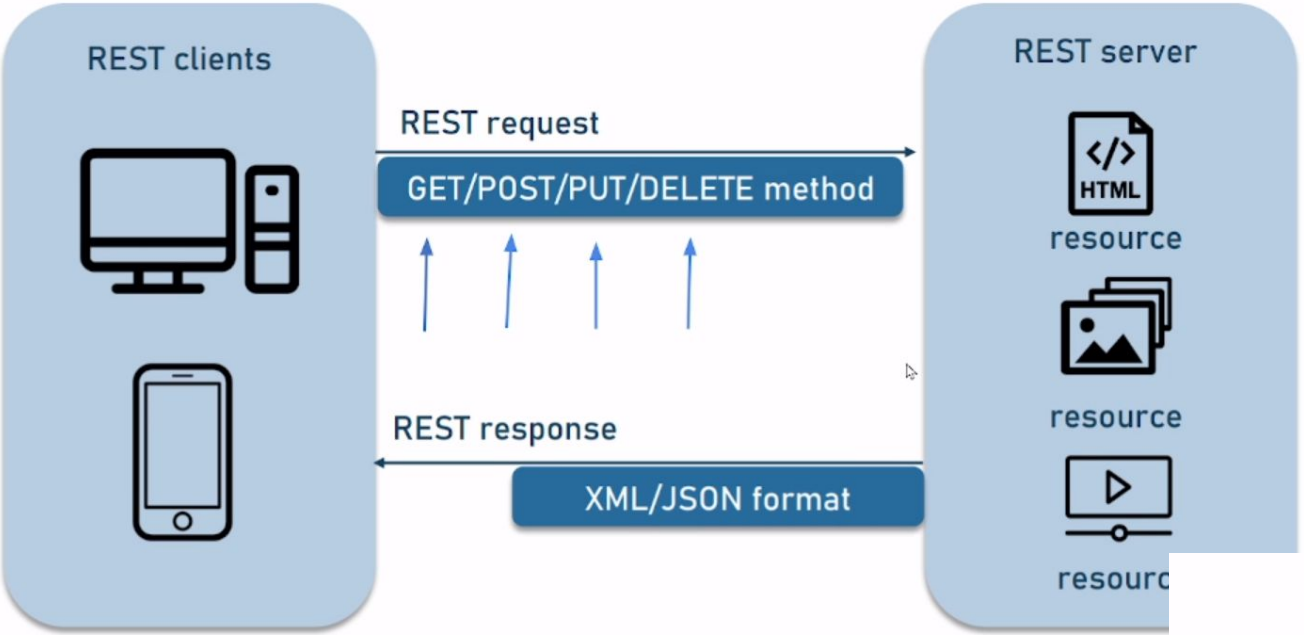
-> Noooooo!!!


API
Client
Server
Database
Database

Rest API in Action

to connect

Client & Server


REST clients
REST request
GET/POST/PUT/DELETE method
REST response
XML/JSON format
REST server
HTML
resource
resource
resourc

Get API =>

সার্ভার এর সাহায্যে **ডাটাবেস থেকে ডাটা** আনবে

Post API =>

সার্ভার এর সাহায্যে **ডাটাবেস এ ডাটা** রাখবে

Put API =>

সার্ভার এর সাহায্যে **ডাটাবেস এর ডাটা** আপডেট করবে

Delete API =>

সার্ভার এর সাহায্যে **ডাটাবেস এর ডাটা** রিমুভ করবে

Let's understand
How POST API connects client & server

বাকি Get, put, delete API
এর জন্য data converstion system একই রকম।

Post API

সার্ভার এর সাহায্যে **ডাটাবেস এ ডাটা রাখবে**

Client Side

Post API

সার্ভার এর সাহায্যে **ডাটাবেস এ ডাটা রাখবে**

```
const coffeeData = { name, chef, supplier,
                     taste, category, details,
                     image, price };

fetch(`http://localhost:5000/coffee`, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(coffeeData),
  })
  .then((res) => res.json())
  .then((data) => {
      console.log(data)
  })
```

Post API

সার্ভার এর সাহায্যে **ডাটাবেস** এ ডাটা **রাখবে**

```
const coffeeData = { name, chef, supplier,
                     taste, category, details,
                     image, price };

fetch(`http://localhost:5000/coffee`, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(coffeeData),
  })
  .then((res) => res.json())
  .then((data) => {
      console.log(data)
  })
```

```
{
  name: 'Americano Coffee',
  chef: 'Mr. Matin Paul',
  supplier: 'Cappu Authorizer',
  taste: 'Sweet and hot',
  category: 'Americano',
  details: 'Espresso with hot water',
  image: 'https://i.ibb.co/PGqMPr9/11.png',
  price: 890
}
```

Post API

সার্ভার এর সাহায্যে **ডাটাবেস এ ডাটা রাখবে**

```
const coffeeData = { name, chef, supplier,
                     taste, category, details,
                     image, price };

fetch(`http://localhost:5000/coffee`, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(coffeeData),
  })
  .then((res) => res.json())
  .then((data) => {
      console.log(data)
  })
```

JSON String

Post API

সার্ভার এর সাহায্যে **ডাটাবেস** এ ডাটা রাখবে

```fetch.js
const coffeeData = { name, chef, supplier,
                     taste, category, details,
                     image, price };

fetch(`http://localhost:5000/coffee`, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(coffeeData),
  })
  .then((res) => res.json())
  .then((data) => {
      console.log(data)
  })
```

```response.log
{
    "name": "Americano Coffee",
    "price": 650,
    "chef": "Mr. Matin Paul",
    "supplier": "Cappu Authorizer",
    "taste": "Sweet and hot",
    "category": "Americano",
    "details": "Espresso with hot water",
    "img": "https://i.ibb.co/PGq
},
```

Post API

সার্ভার এর সাহায্যে **ডাটাবেস এ ডাটা রাখবে**

Server Side

Post API

সার্ভার এর সাহায্যে **ডাটাবেস এ ডাটা রাখবে**

```
app.use(express.json());
```

Post API

সার্ভার এর সাহায্যে **ডাটাবেস এ ডাটা রাখবে**

```
app.use(express.json());
```

→ JSON String

Post API

সার্ভার এর সাহায্যে ডাটাবেস এ ডাটা রাখবে

```
// add a coffee
app.post("/coffee", async (req, res) => {
   const data = req.body;
   const result = await coffeeCollection.insertOne(data);
   res.send(result);
});
```

Post API

সার্ভার এর সাহায্যে **ডাটাবেস এ ডাটা রাখবে**

```
// add a coffee
app.post("/coffee", async (req, res) => {
   const data = req.body;
   const result = await coffeeCollection.insertOne(data);
   res.send(result);
});
```

Response from server
After inserting data
into database

Post API

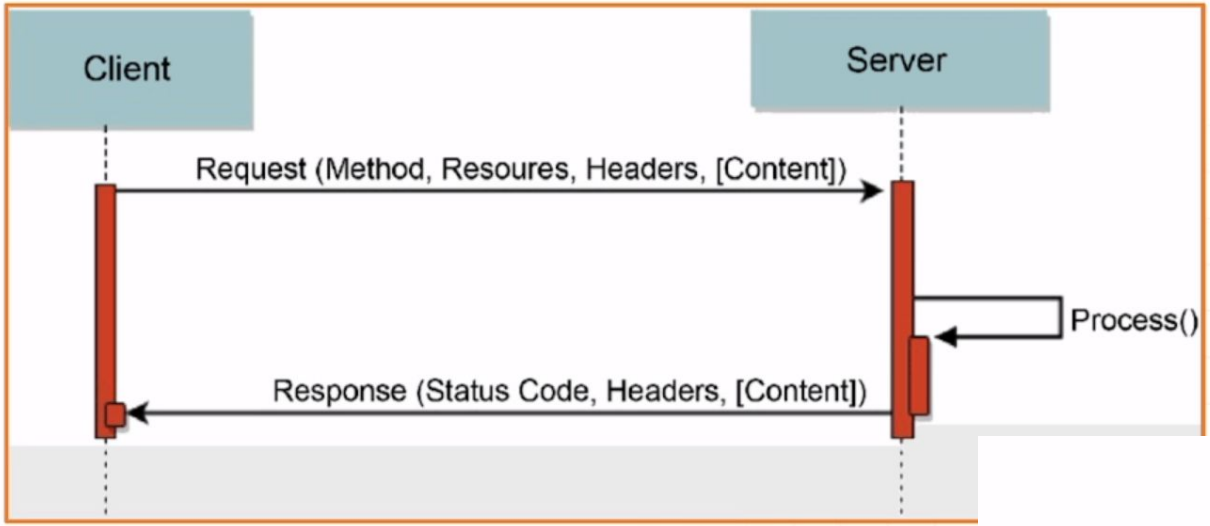
সার্ভার এর সাহায্যে **ডাটাবেস** এ ডাটা রাখবে

```
// add a coffee
app.post("/coffee", async (req, res) => {
   const data = req.body;
   const result = await coffeeCollection.insertOne(data);
   res.send(result);
});
```

```
{
  acknowledged: true,
  insertedId: new ObjectId("642fdbf802002b9689d82c4c")
}
```

Request Response

Request/ Response Model



Request (req) object

The req object represents the HTTP request and has properties

- request query string,
- parameters,
- body,
- HTTP headers, and so on

req.body

Contains key-value pairs of data submitted in the request body

```
app.post('/profile', function (req, res) {
  console.log(req.body)
  res.json(req.body)
})
```

req.params

This property is an object containing properties mapped to the named route “parameters”. For example, if you have the route /user/:name, then the “name” property is available as req.params.name. This object defaults to {}.

```
// GET /user/tj
console.dir(req.params.name)
// => 'tj'
```

req.query

This property is an object containing a property for each query string parameter in the route. When query parser is set to disabled, it is an empty object {}, otherwise it is the result of the configured query parser.

Response (res) object

The res object represents the HTTP response that an Express app sends when it gets an HTTP request and has methods

- res.send(),
- res.json(),
- res.status(), res.sendStatus(),
- res.set(), and so on

res.send()

Sends the HTTP response.

The body parameter can be a Buffer object, a String, an object, Boolean, or an Array.

```
res.send({ some: 'json' })
res.send('<p>some html</p>')
```

res.json()

Sends a JSON response. This method sends a response (with the correct content-type) that is the parameter converted to a JSON string using JSON.stringify().

The parameter can be any JSON type, including object, array, string, Boolean, number, or null, and you can also use it to convert other values to JSON.

```
res.json(null)
res.json({ user: 'tobi' })
```

res.status()

Sets the HTTP status for the response.

```
res.status(403).end()
res.status(400).send('Bad Request')
res.status(404).sendFile('/absolute/path/to/404.png')
```

res.sendStatus()

Sets the response HTTP status code to statusCode and sends the registered status message as the text response body. If an unknown status code is specified, the response body will just be the code number.

```
res.sendStatus(404)
```

res.set()

Sets the response's HTTP header field to value. To set multiple fields at once, pass an object as the parameter.

```
res.set('Content-Type', 'text/plain')

res.set({
  'Content-Type': 'text/plain',
  'Content-Length': '123',
  ETag: '12345'
})
```

Thank you!