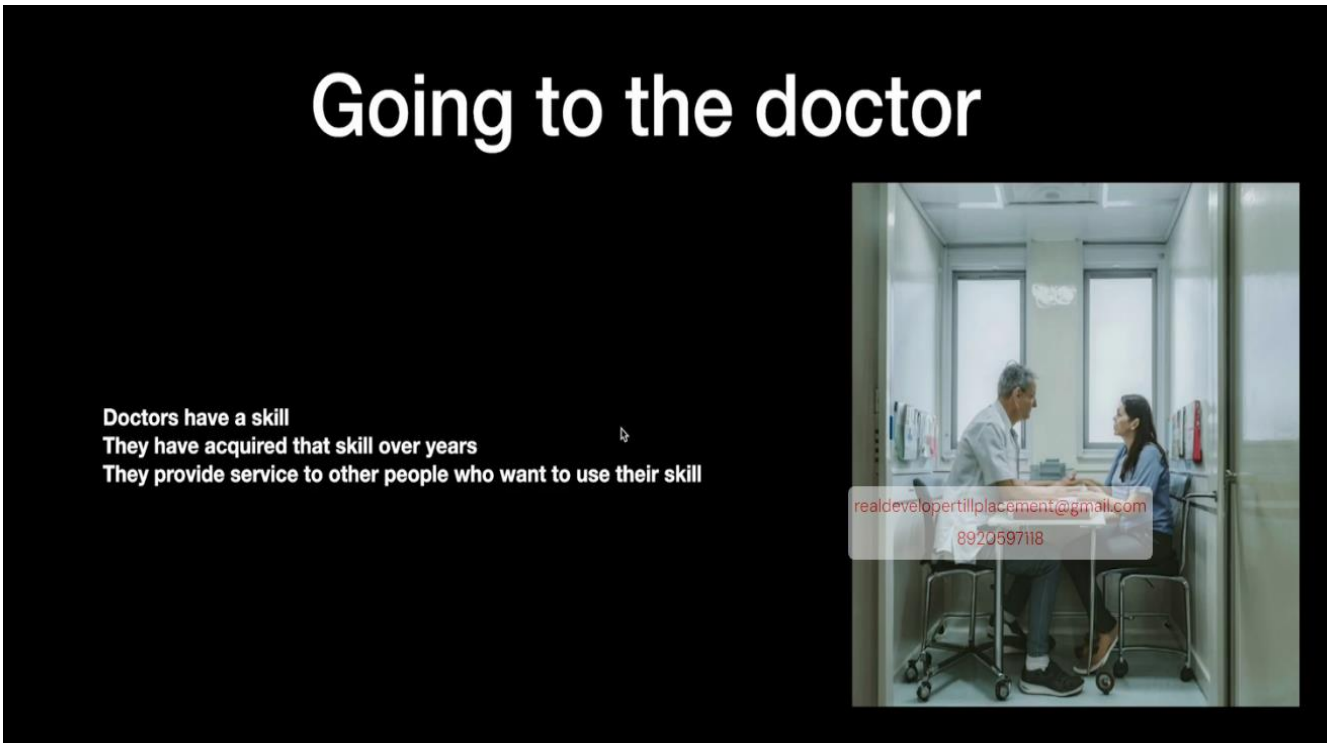
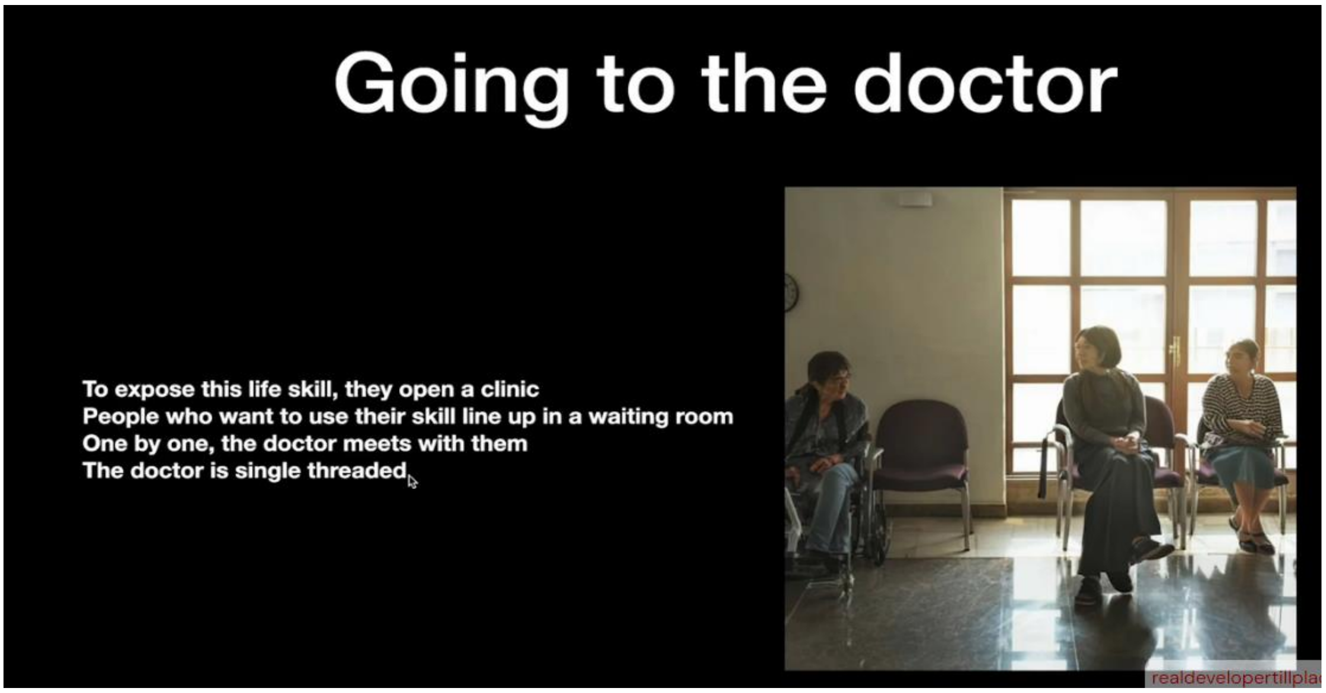
EXPRESS BASICS



Going to the doctor

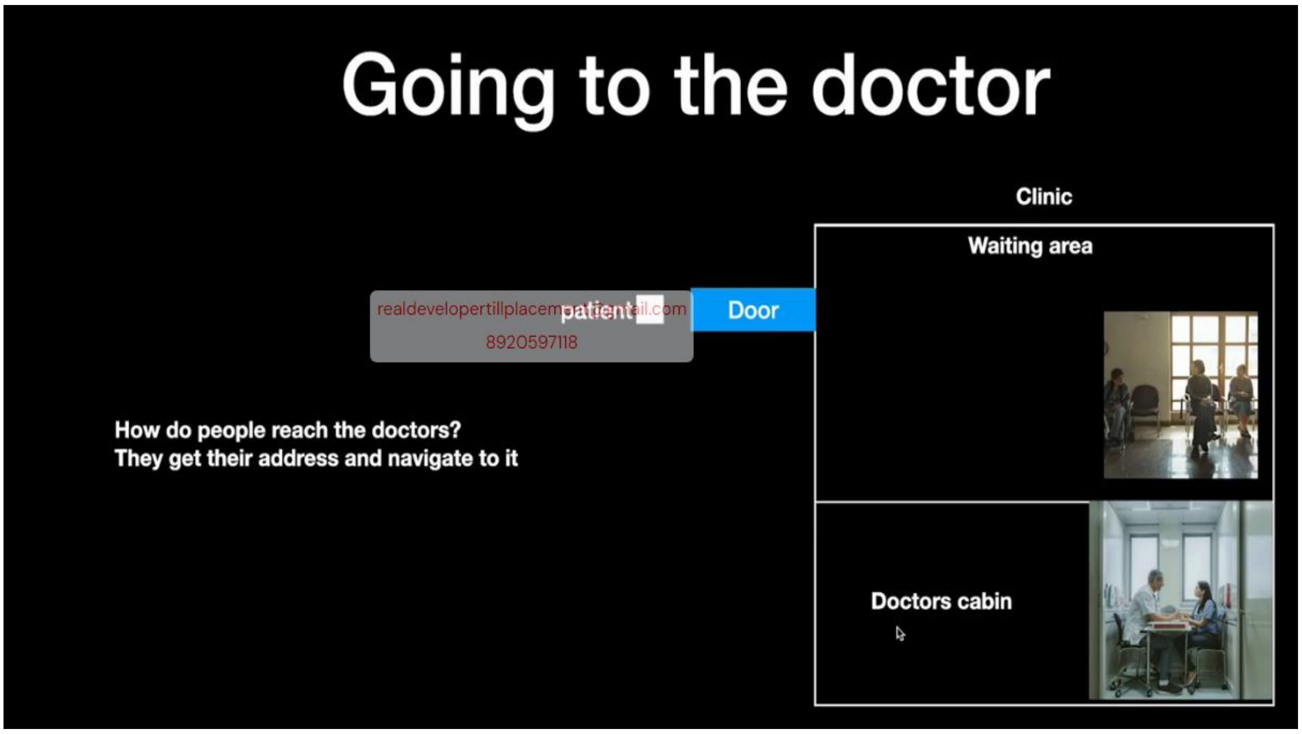
Doctors have a skill
They have acquired that skill over years
They provide service to other people who want to use their skill



Going to the doctor

To expose this life skill, they open a clinic
People who want to use their skill line up in a waiting room
One by one, the doctor meets with them
The doctor is single threaded

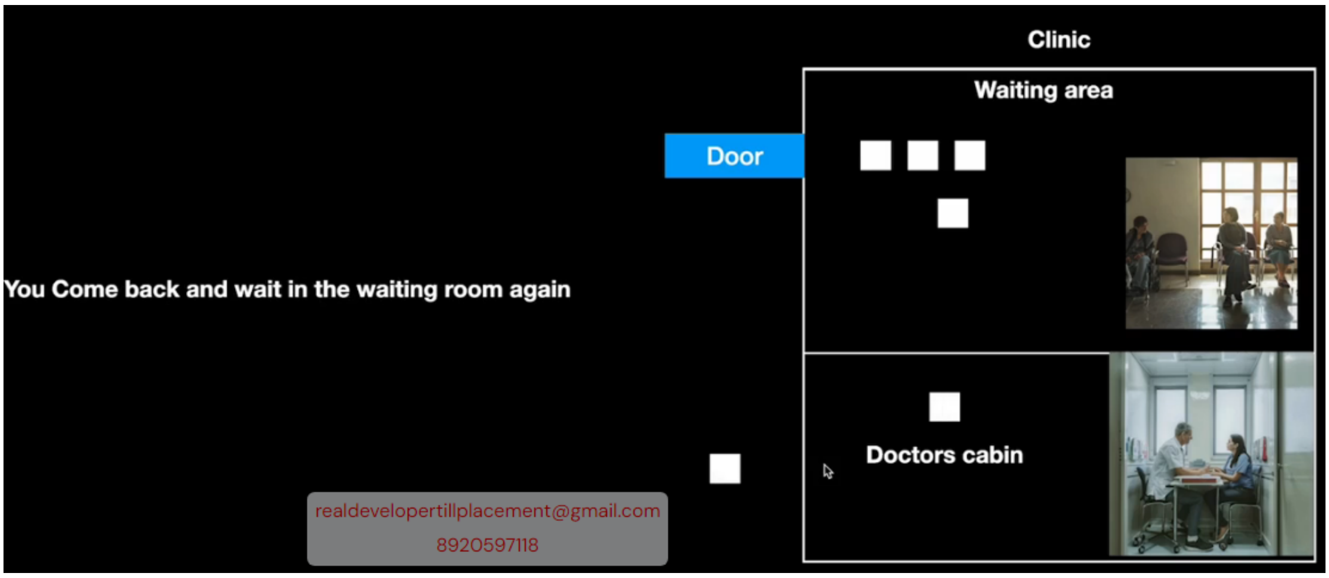
the doctor is single threaded means the doctor tackles one patient at a time.



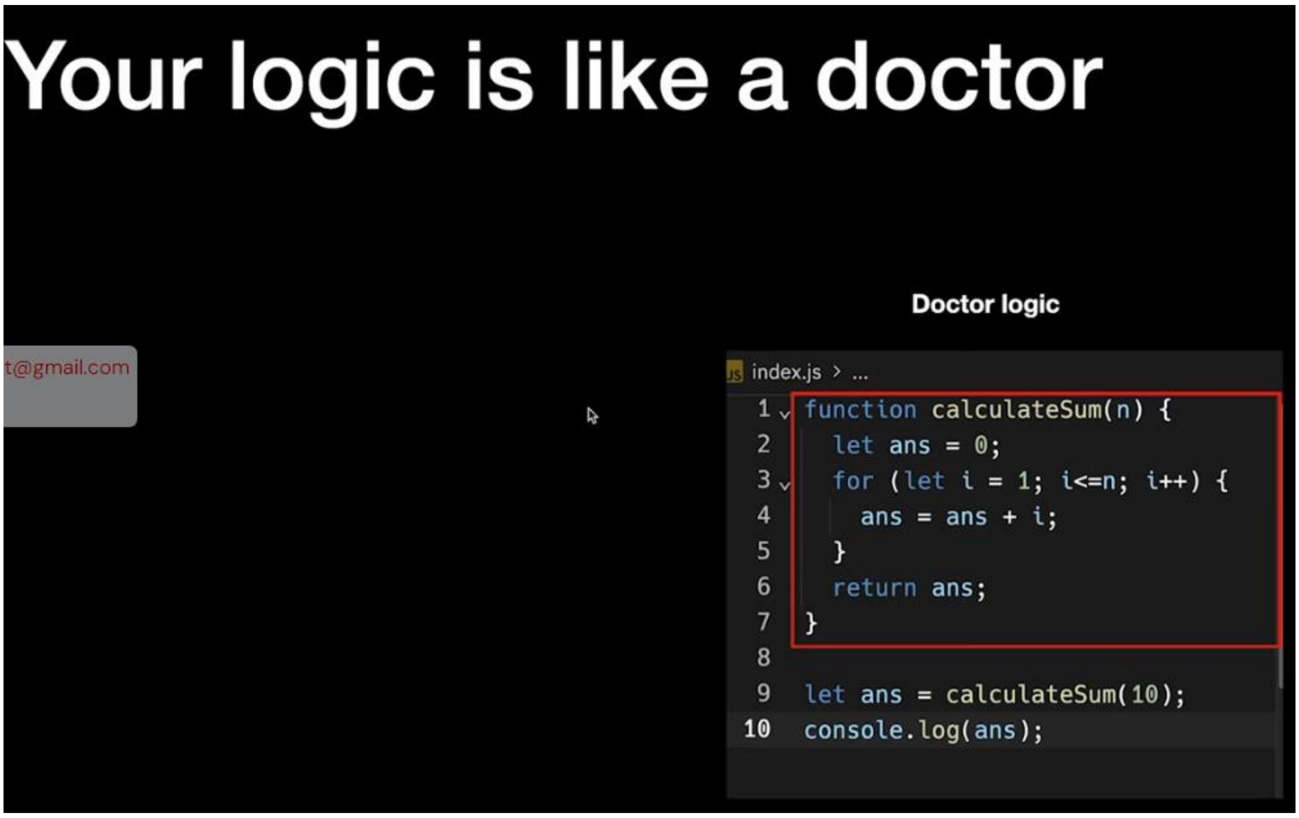
Going to the doctor

How do people reach the doctors?
They get their address and navigate to it

Patients generally enter the waiting area, to meet the doctor, but when they reach the doctor, the doctor tells them to buy some medicines and meanwhile doctor attends to another patient. This is somewhat kind of an Asynchronous function.


Clinic
Waiting area
Door
You Come back and wait in the waiting room again
Doctors cabin
realdevelopertillplacement@gmail.com
8920597118

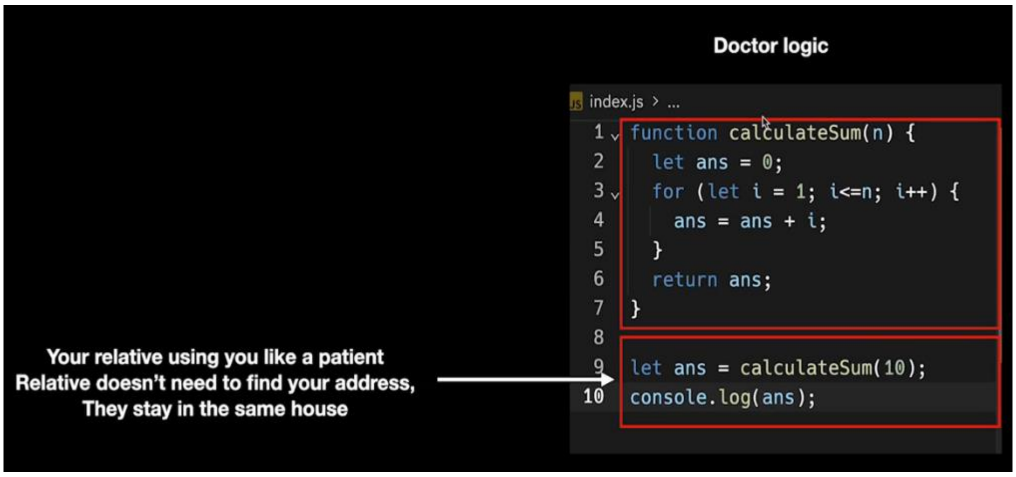
The doctor is similar to our logic.



Your logic is like a doctor

```
function calculateSum(n) {
  let ans = 0;
  for (let i = 1; i<=n; i++) {
    ans = ans + i;
  }
  return ans;
}

let ans = calculateSum(10);
console.log(ans);
```

We have this logic on the same page (index.js). like a family has a doctor, so the patients in the family do not need to find the address of the doctor, they all stay in the same house.

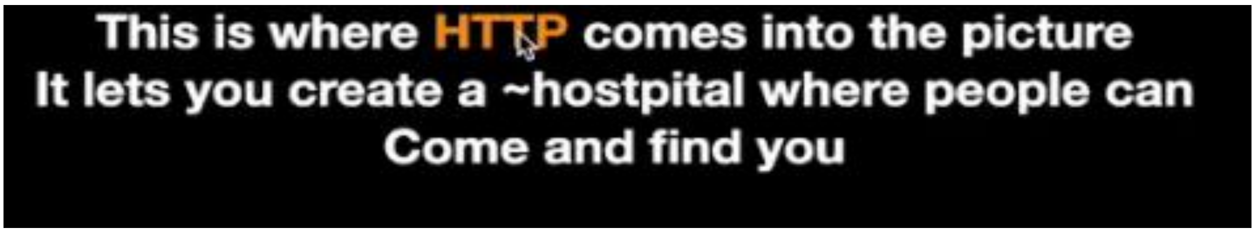

Doctor logic

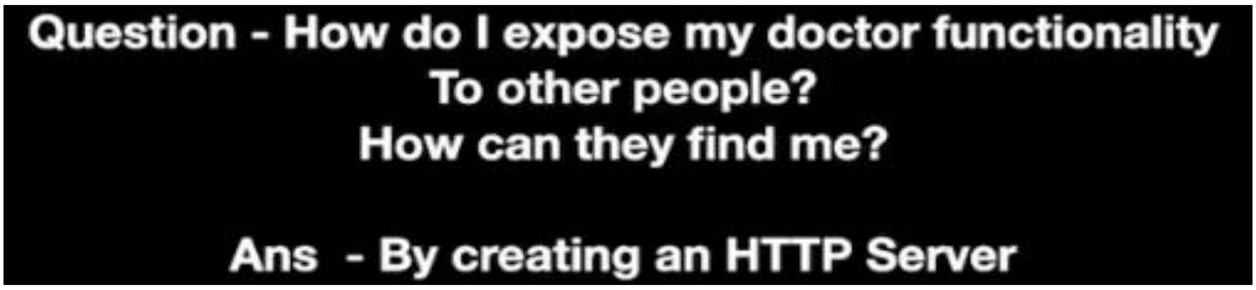
index.js > ...

```js
function calculateSum(n) {
  let ans = 0;
  for (let i = 1; i<=n; i++) {
    ans = ans + i;
  }
  return ans;
}

let ans = calculateSum(10);
console.log(ans);
```

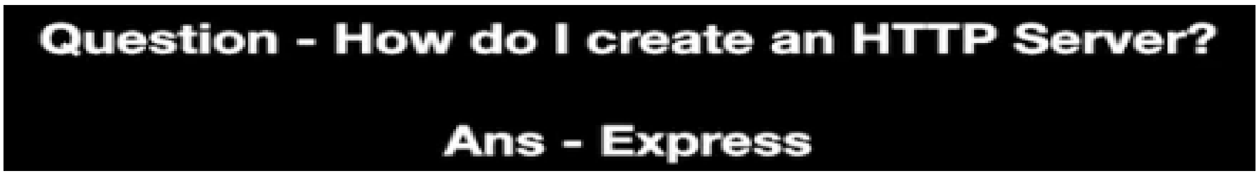
Your relative using you like a patient
Relative doesn't need to find your address,
They stay in the same house

But how do we expose our logic to the world?


This is where HTTP comes into the picture
It lets you create a ~hostpital where people can
Come and find you

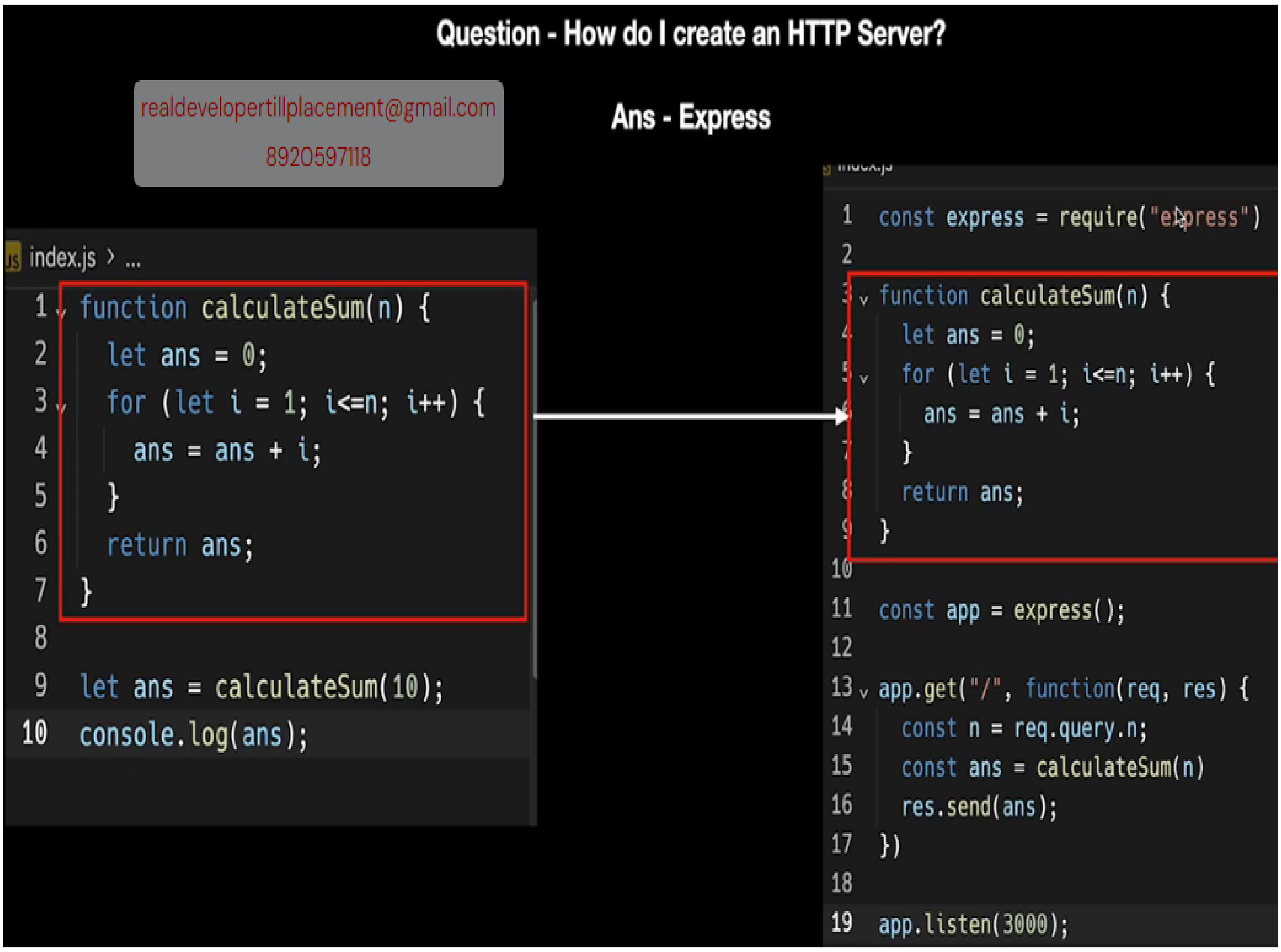

Question - How do I expose my doctor functionality
To other people?
How can they find me?

Ans - By creating an HTTP Server


Question - How do I create an HTTP Server?

Ans - Express

A server is like a hospital where the doctor (our logic) is there, and others can come a fetch the logic or see the doctor.



Question - How do I create an HTTP Server?

realdevelopertillplacement@gmail.com
8920597118

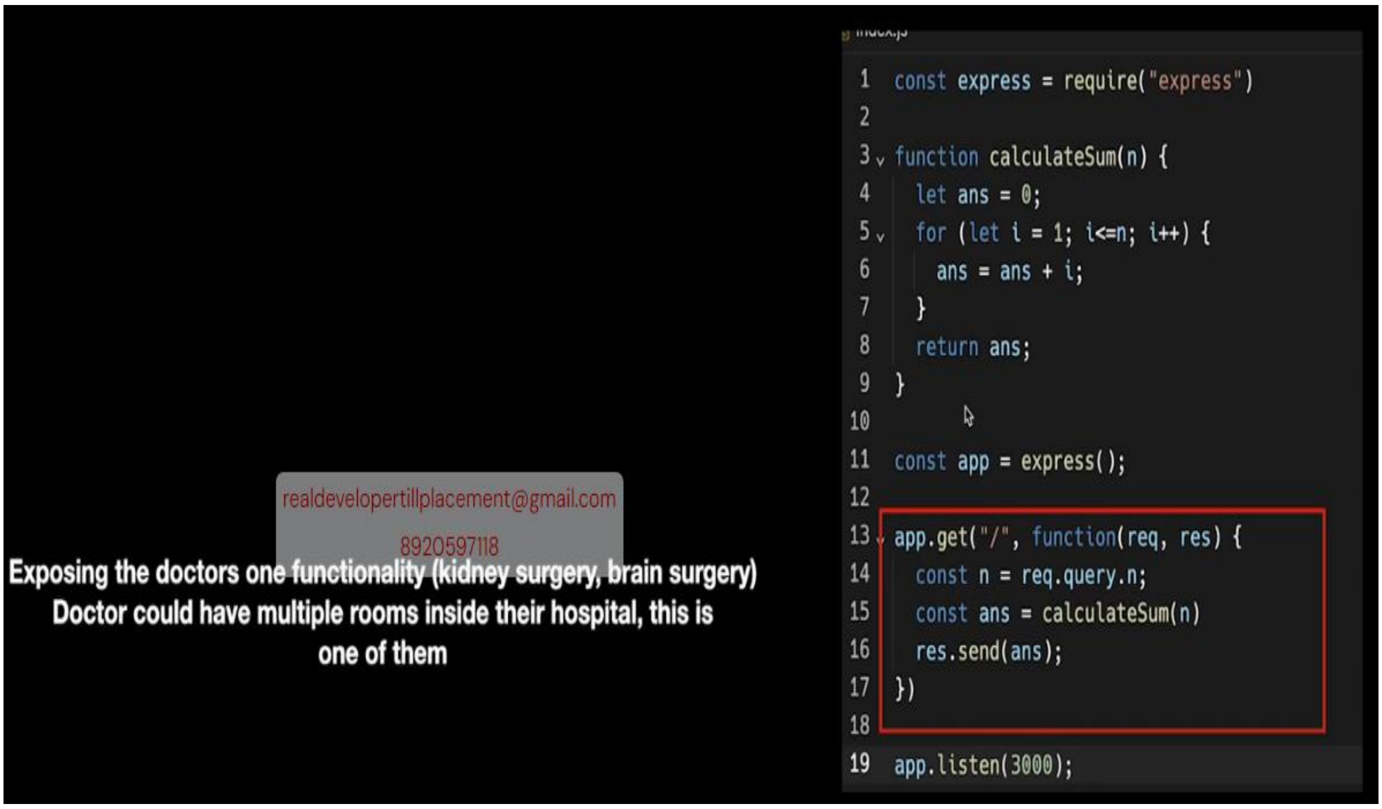
Ans - Express

index.js > ...

```
function calculateSum(n) {
  let ans = 0;
  for (let i = 1; i<=n; i++) {
    ans = ans + i;
  }
  return ans;
}

let ans = calculateSum(10);
console.log(ans);
```

index.js

```
const express = require("express")

function calculateSum(n) {
  let ans = 0;
  for (let i = 1; i<=n; i++) {
    ans = ans + i;
  }
  return ans;
}

const app = express();

app.get("/", function(req, res) {
  const n = req.query.n;
  const ans = calculateSum(n)
  res.send(ans);
})

app.listen(3000);
```

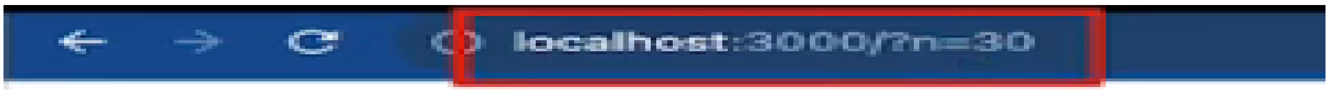


Exposing the doctors one functionality (kidney surgery, brain surgery)
Doctor could have multiple rooms inside their hospital, this is one of them

```
index.js
const express = require("express")

function calculateSum(n) {
  let ans = 0;
  for (let i = 1; i<=n; i++) {
    ans = ans + i;
  }
  return ans;
}

const app = express();

app.get("/", function(req, res) {
  const n = req.query.n;
  const ans = calculateSum(n)
  res.send(ans);
})

app.listen(3000);
```

app.listen(3000);, it decides the address of the clinic.

App.get(); decides which doctor to show according to the need of the patient (the URL which we write in the browser according to that page renders).

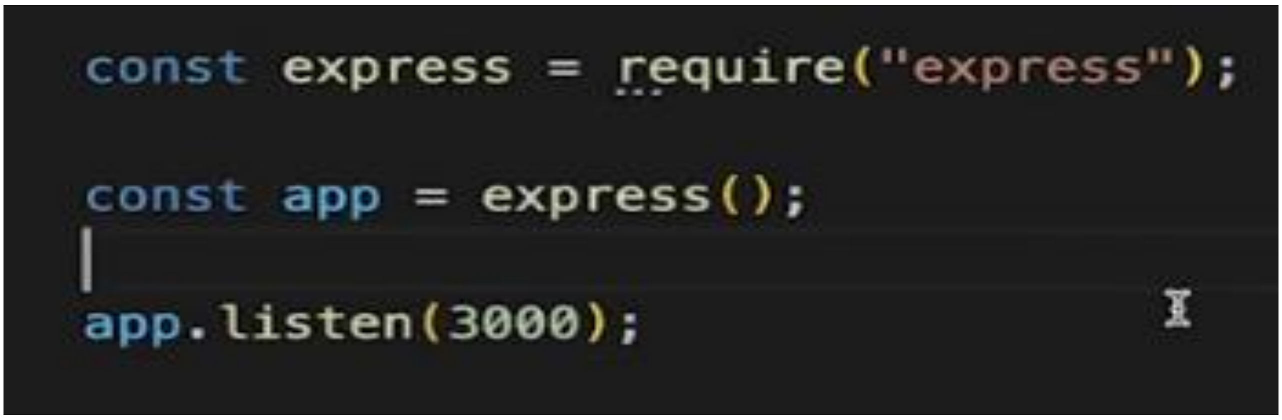


```
localhost:3000/?n=30
```

This is how the request is being made where the query parameters are written after the question mark.

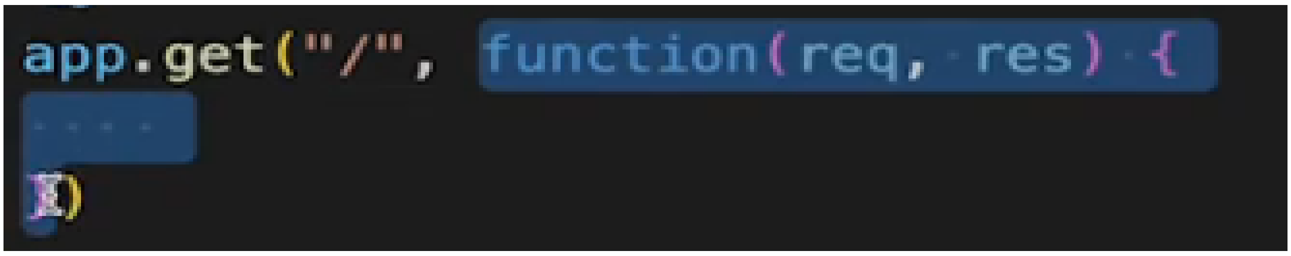
const app=express(); is like declaring a hospital/clinic, like we have created the hospital now patients can come.

The basic 3 lines of code we need to start a server.



```
const express = require("express");

const app = express();

app.listen(3000);
```

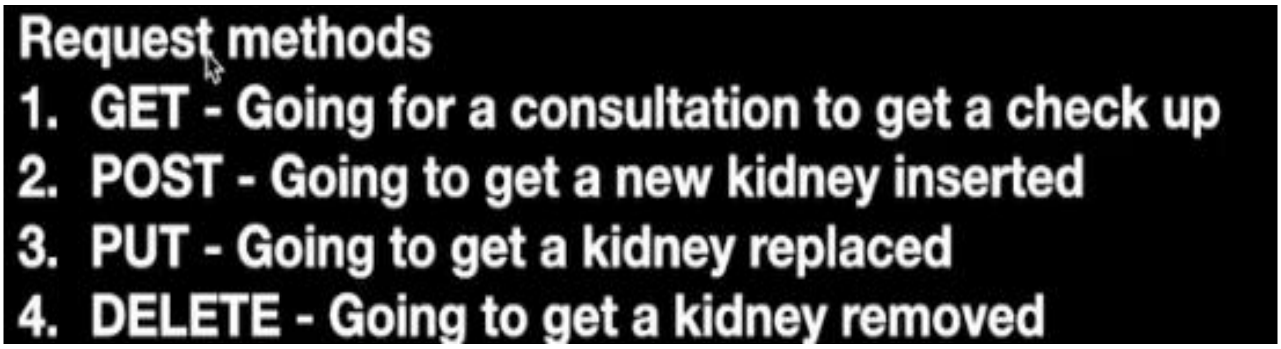
Now we will see how the get function works:-



```
app.get("/", function(req, res) {

})
```

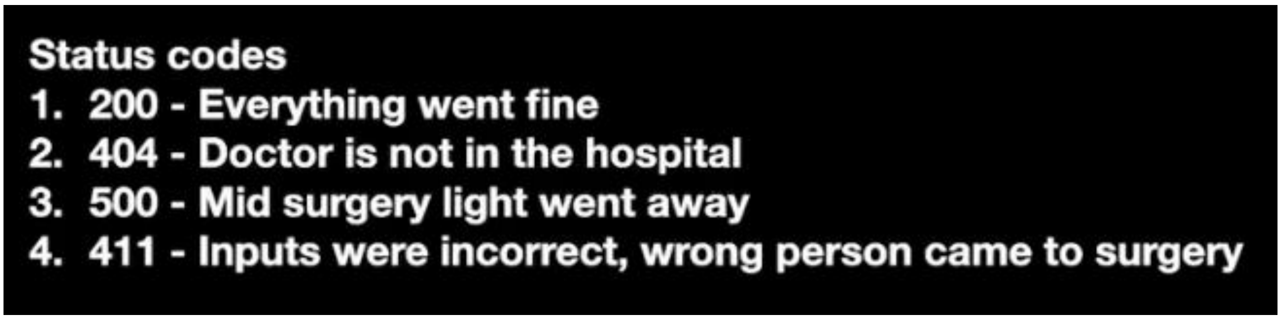
The 1st argument is the URL (If we search this URL on Google then we get the data, written inside the get function). The 2nd argument is the callback function, which means any patient waiting inside the waiting room, sends the patient inside the doctor's room (this callback function is called whenever the person comes on the browser).

There are some request methods:-



Request methods
1. GET - Going for a consultation to get a check up
2. POST - Going to get a new kidney inserted
3. PUT - Going to get a kidney replaced
4. DELETE - Going to get a kidney removed

Some Status Codes:-



Status codes
1. 200 - Everything went fine
2. 404 - Doctor is not in the hospital
3. 500 - Mid surgery light went away
4. 411 - Inputs were incorrect, wrong person came to surgery

Informational responses (1xx): These status codes indicate that the request has been received and understood and that the process is continuing.

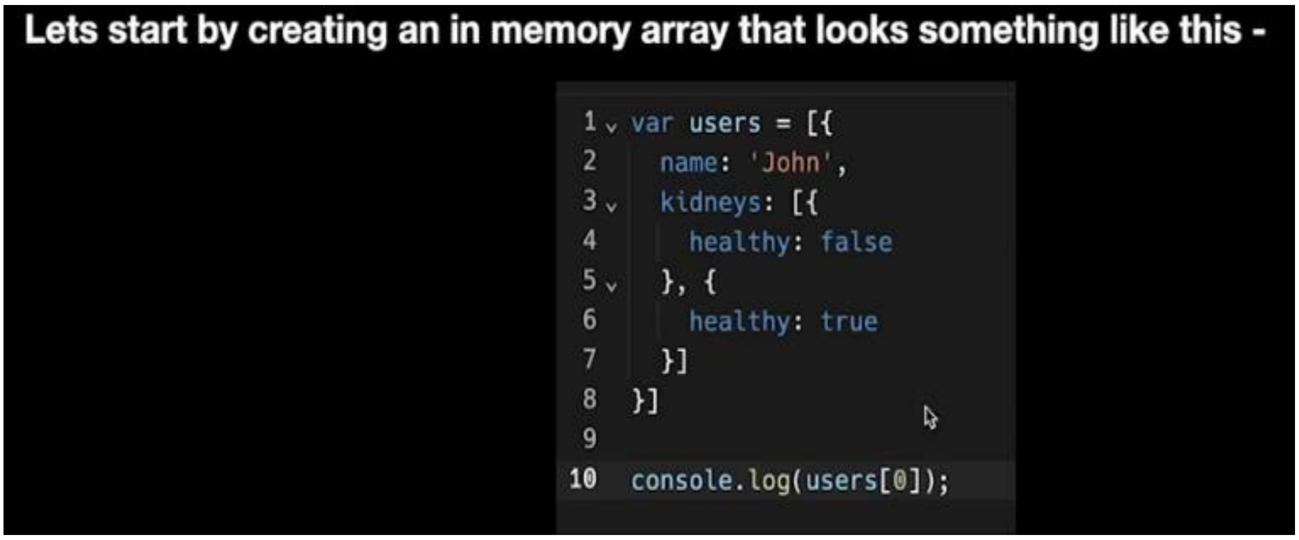
Successful responses (2xx): These status codes indicate that the request was successfully received, understood, and accepted.

Redirection messages (3xx): These status codes indicate that further action needs to be taken by the client to fulfill the request.

Client error responses (4xx): These status codes indicate that the client seems to have made a mistake in the request, and the server cannot process it.

Server error responses (5xx): These status codes indicate that the server failed to fulfill a valid request due to some error on its end.

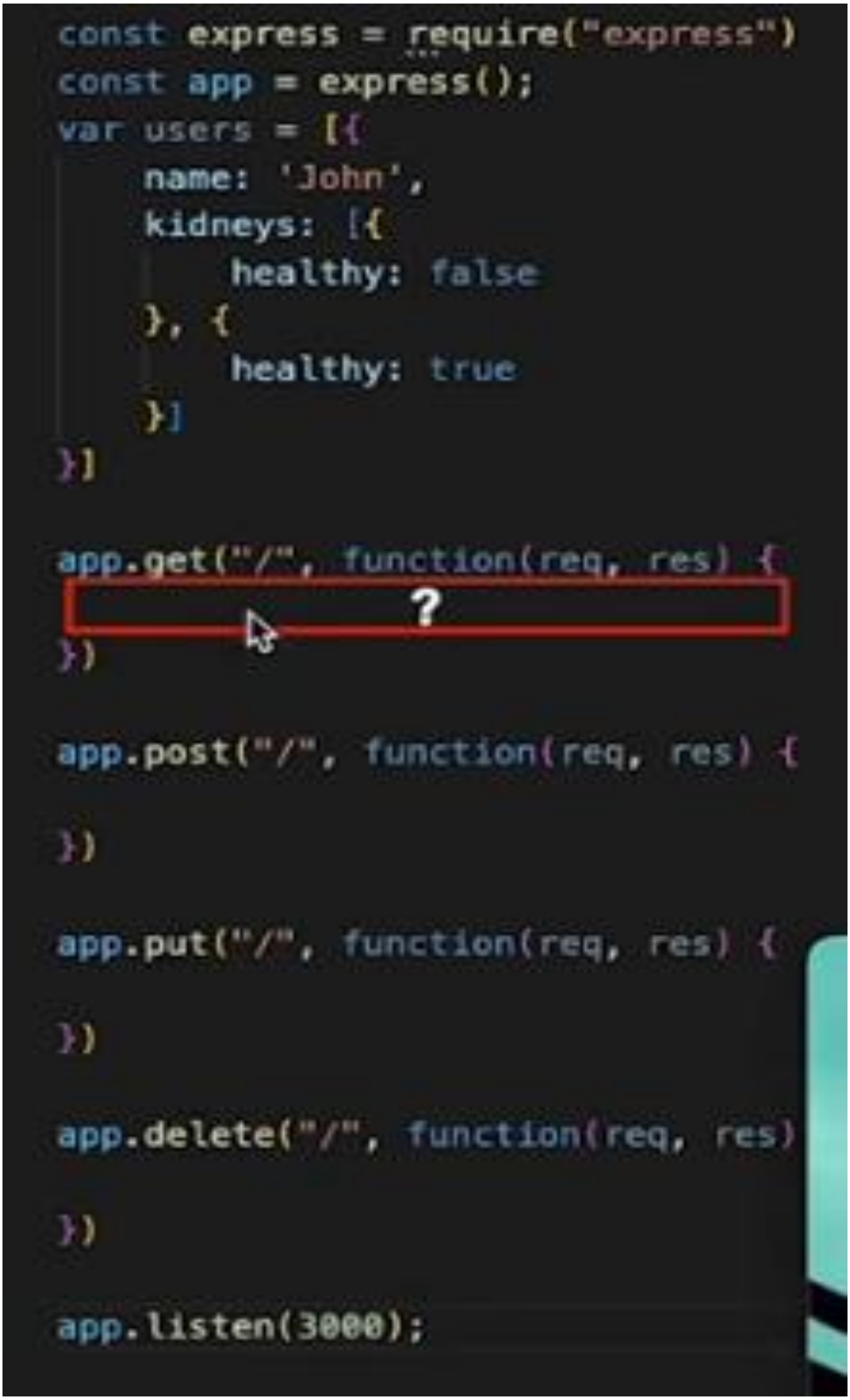
Let's learn the post methods and status codes by creating the in-memory database(our own dummy database).



Lets start by creating an in memory array that looks something like this -

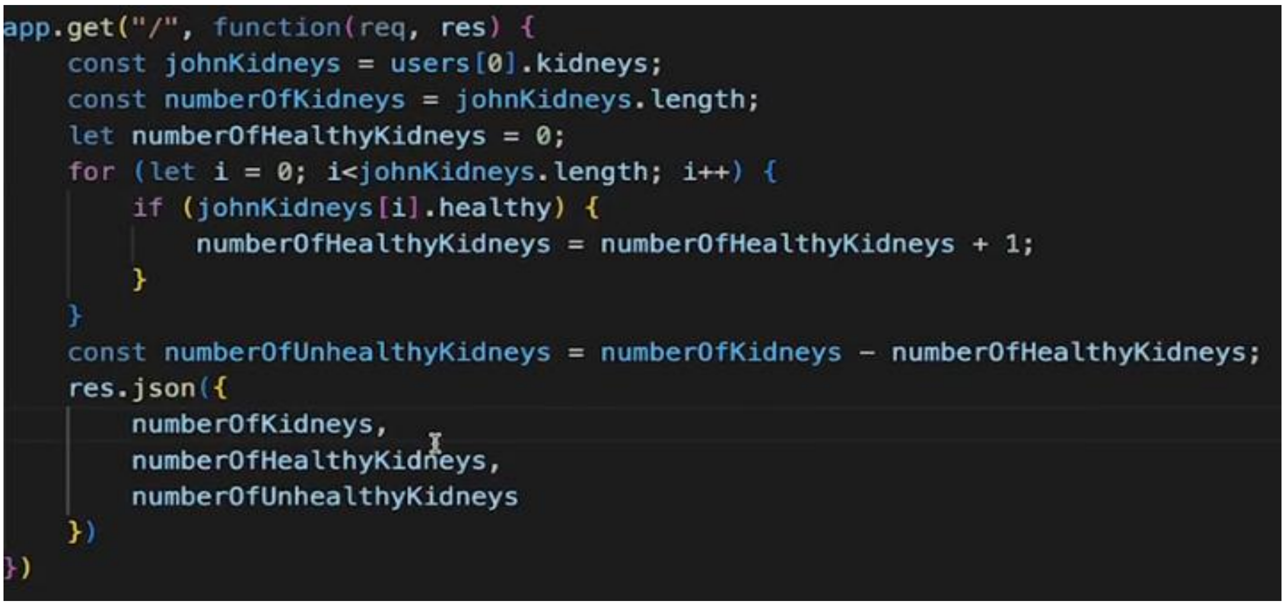
```
var users = [{
  name: 'John',
  kidneys: [{
    healthy: false
  }, {
    healthy: true
  }]
}]

console.log(users[0]);
```

Let’s have a boiler plate:-



```
const express = require("express")
const app = express();
var users = [{
    name: 'John',
    kidneys: [{
        healthy: false
    }, {
        healthy: true
    }]
}]

app.get("/", function(req, res) {
                    ?
})

app.post("/", function(req, res) {

})

app.put("/", function(req, res) {

})

app.delete("/", function(req, res)

})

app.listen(3000);
```

First, we have written for, the GET Method:-

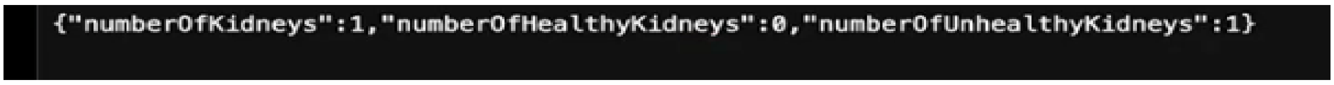


```
app.get("/", function(req, res) {
    const johnKidneys = users[0].kidneys;
    const numberOfKidneys = johnKidneys.length;
    let numberOfHealthyKidneys = 0;
    for (let i = 0; i<johnKidneys.length; i++) {
        if (johnKidneys[i].healthy) {
            numberOfHealthyKidneys = numberOfHealthyKidneys + 1;
        }
    }
    const numberOfUnhealthyKidneys = numberOfKidneys - numberOfHealthyKidneys;
    res.json({
        numberOfKidneys,
        numberOfHealthyKidneys,
        numberOfUnhealthyKidneys
    })
})
```

Res.json() will send the by converting the data into

key-value pairs.

Like this:-

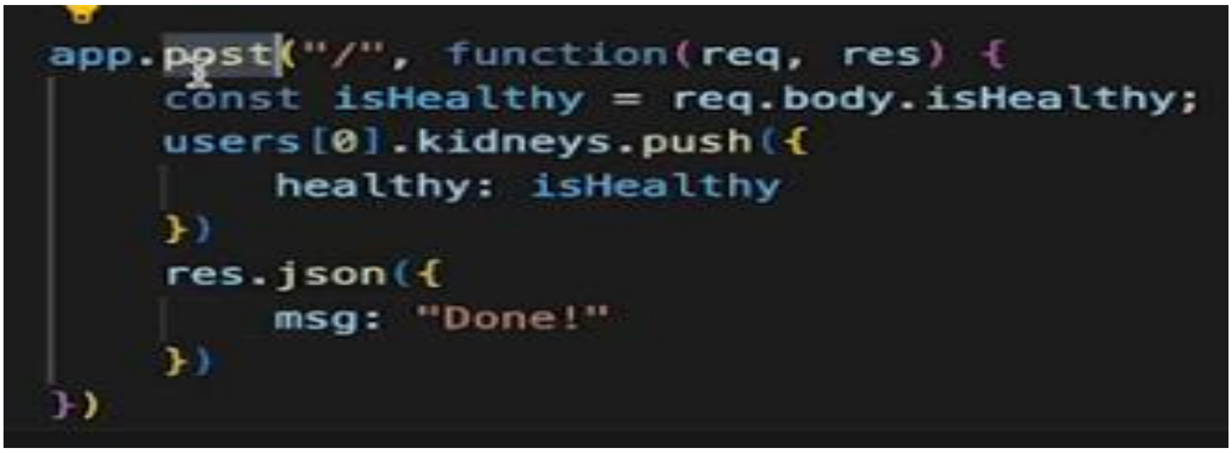


```
{"numberOfKidneys":1,"numberOfHealthyKidneys":0,"numberOfUnhealthyKidneys":1}
```

Now we are writing for POST Method:-

By using the post method we can add our data to

the body of the database.

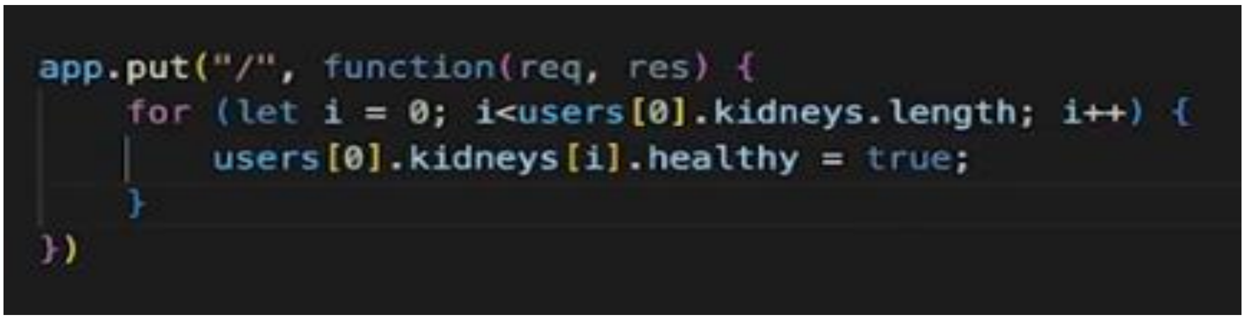


```
app.post("/", function(req, res) {
    const isHealthy = req.body.isHealthy;
    users[0].kidneys.push({
        healthy: isHealthy
    })
    res.json({
        msg: "Done!"
    })
})
```

We are pushing new kidneys to the user with a status of healthy.

For PUT Method:-

By using the PUT method we can update our database.

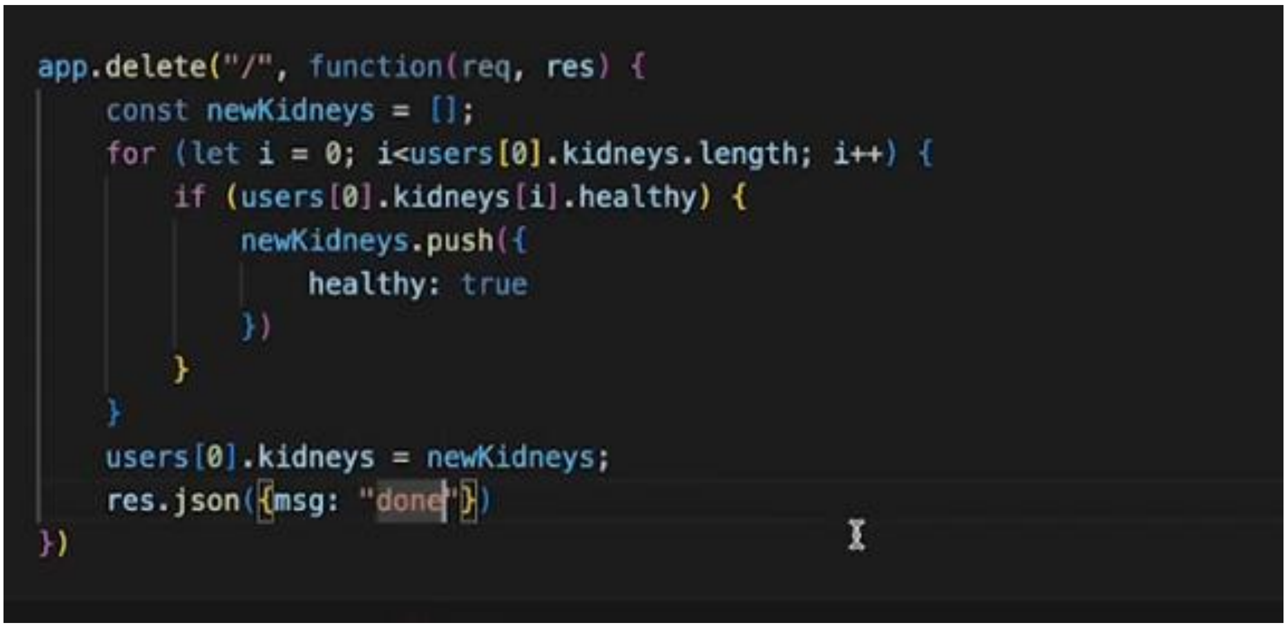


```
app.put("/", function(req, res) {
    for (let i = 0; i<users[0].kidneys.length; i++) {
        users[0].kidneys[i].healthy = true;
    }
})
```

After for loop write res.json({});

For DELETE Method:-

By using this method we can delete some data from our database.



```
app.delete("/", function(req, res) {
    const newKidneys = [];
    for (let i = 0; i<users[0].kidneys.length; i++) {
        if (users[0].kidneys[i].healthy) {
            newKidneys.push({
                healthy: true
            })
        }
    }
    users[0].kidneys = newKidneys;
    res.json({msg: "done"})
})
```