Express and HTTP with real world examples
Postman

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

Going to the doctor

patient

Door

Clinic

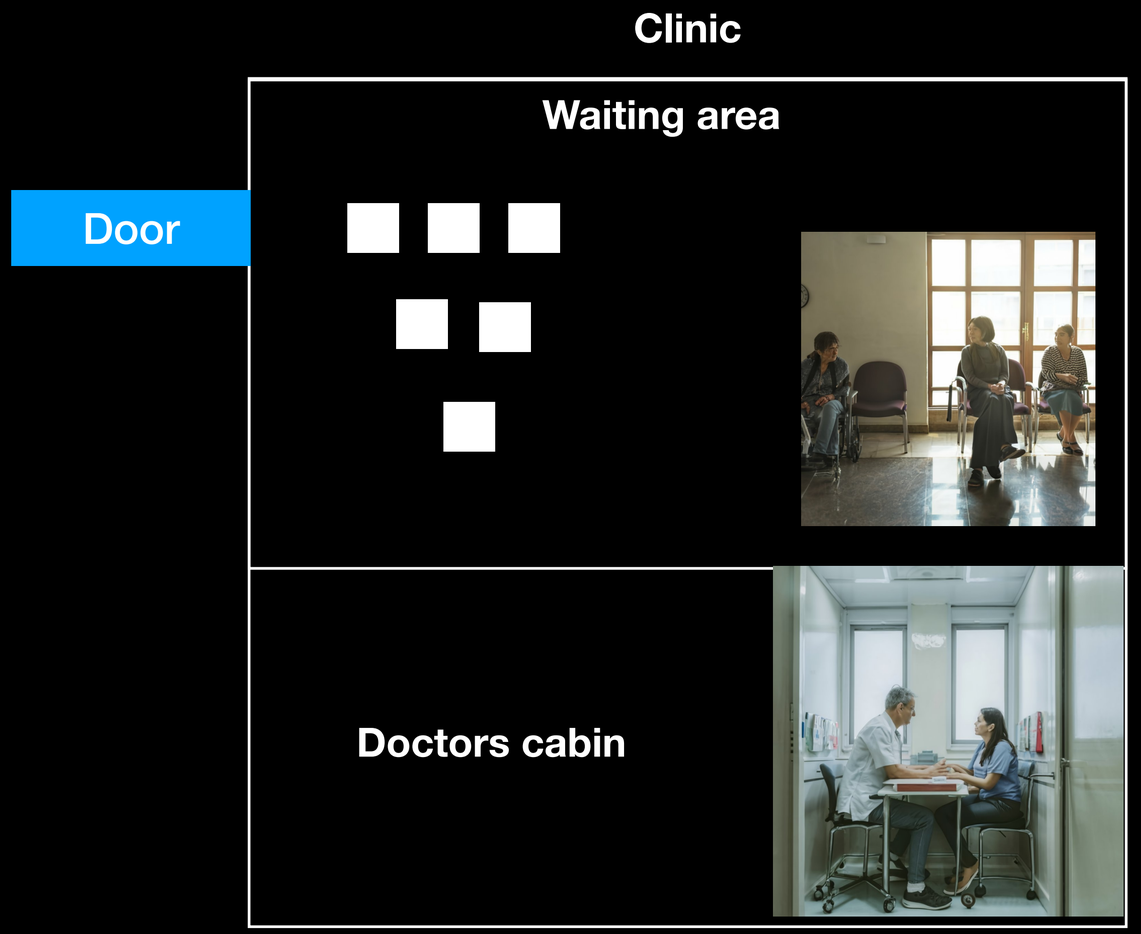
Waiting area

Doctors cabin

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

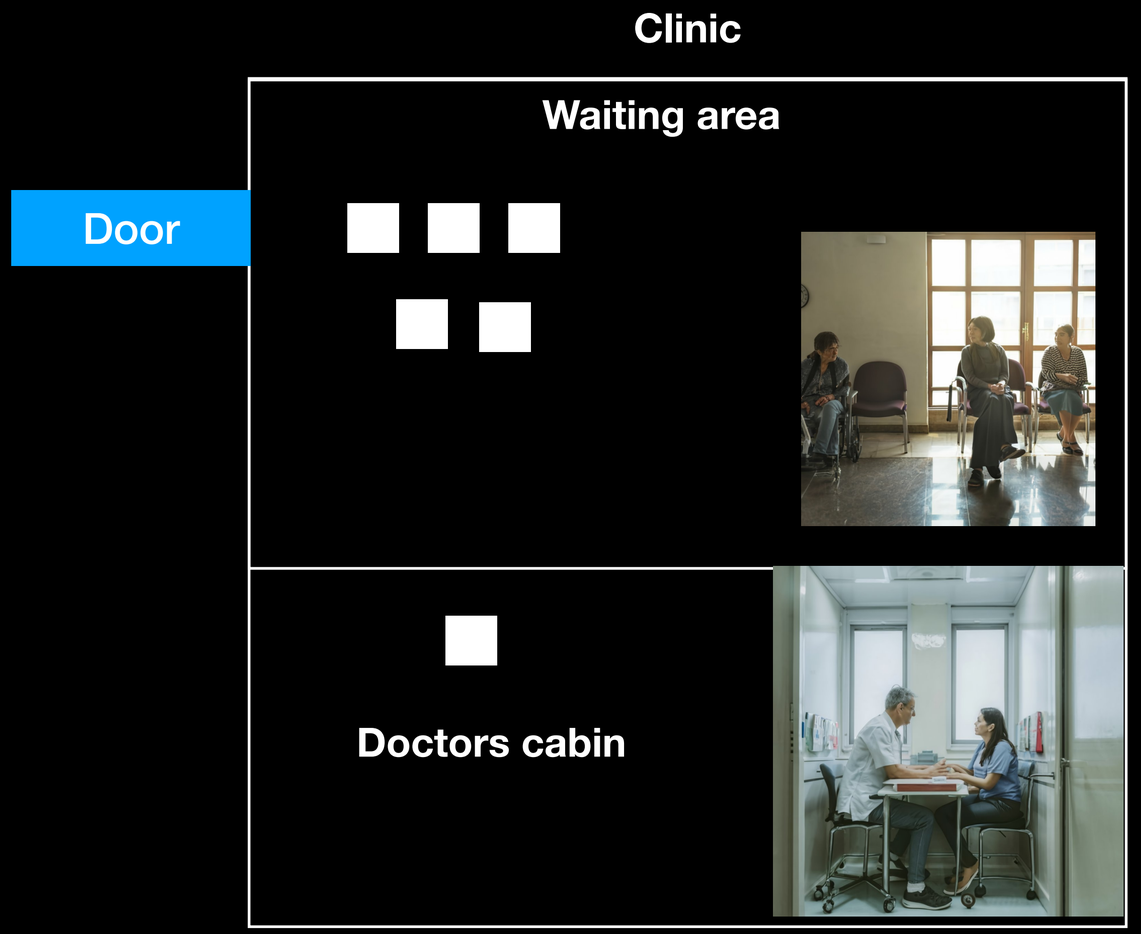
Going to the doctor

Once they reach there, they wait in the waiting area
Until their time comes



Going to the doctor

Doctor tends to them one by one



Going to the doctor

Clinic

Waiting area

Door

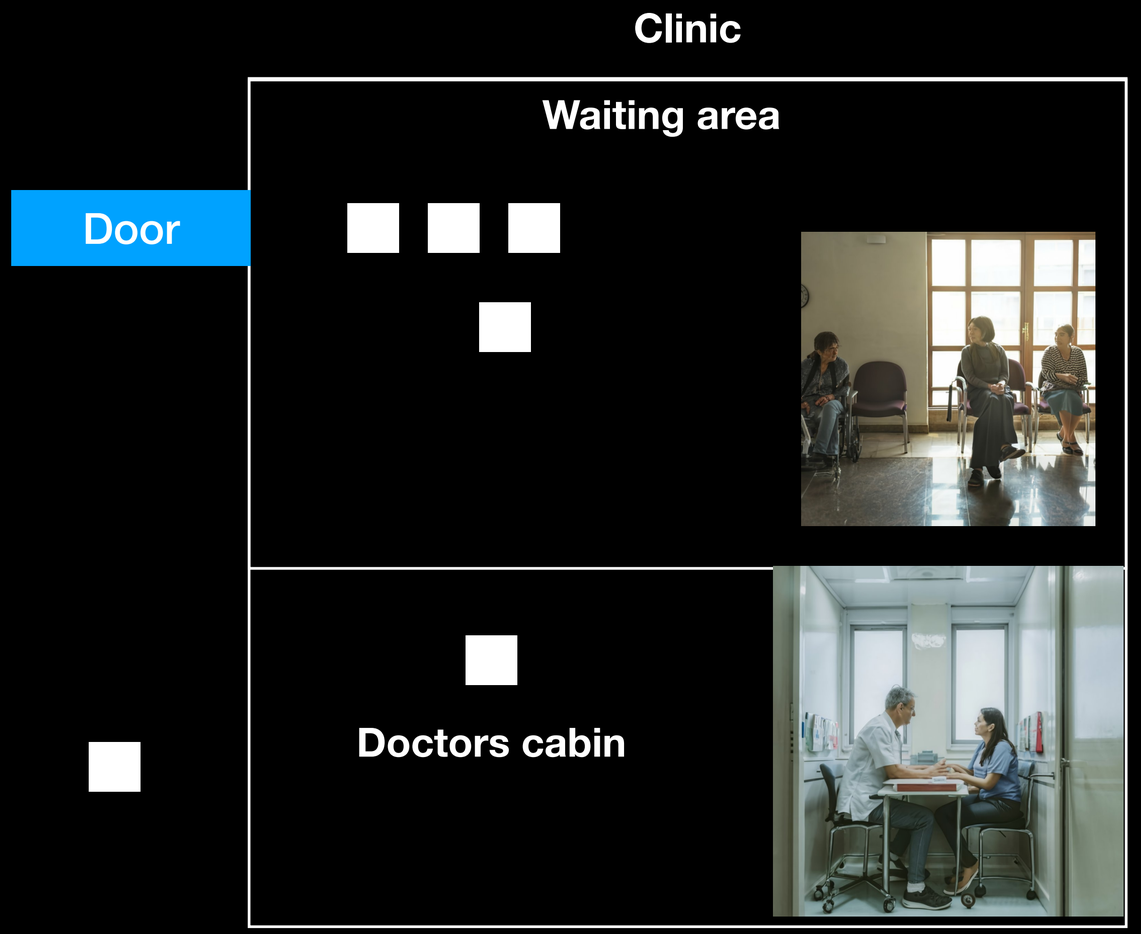
Doctor can tell them to get a medicine in the middle and meanwhile tend to other people

Doctors cabin

Go get medicine from chemist

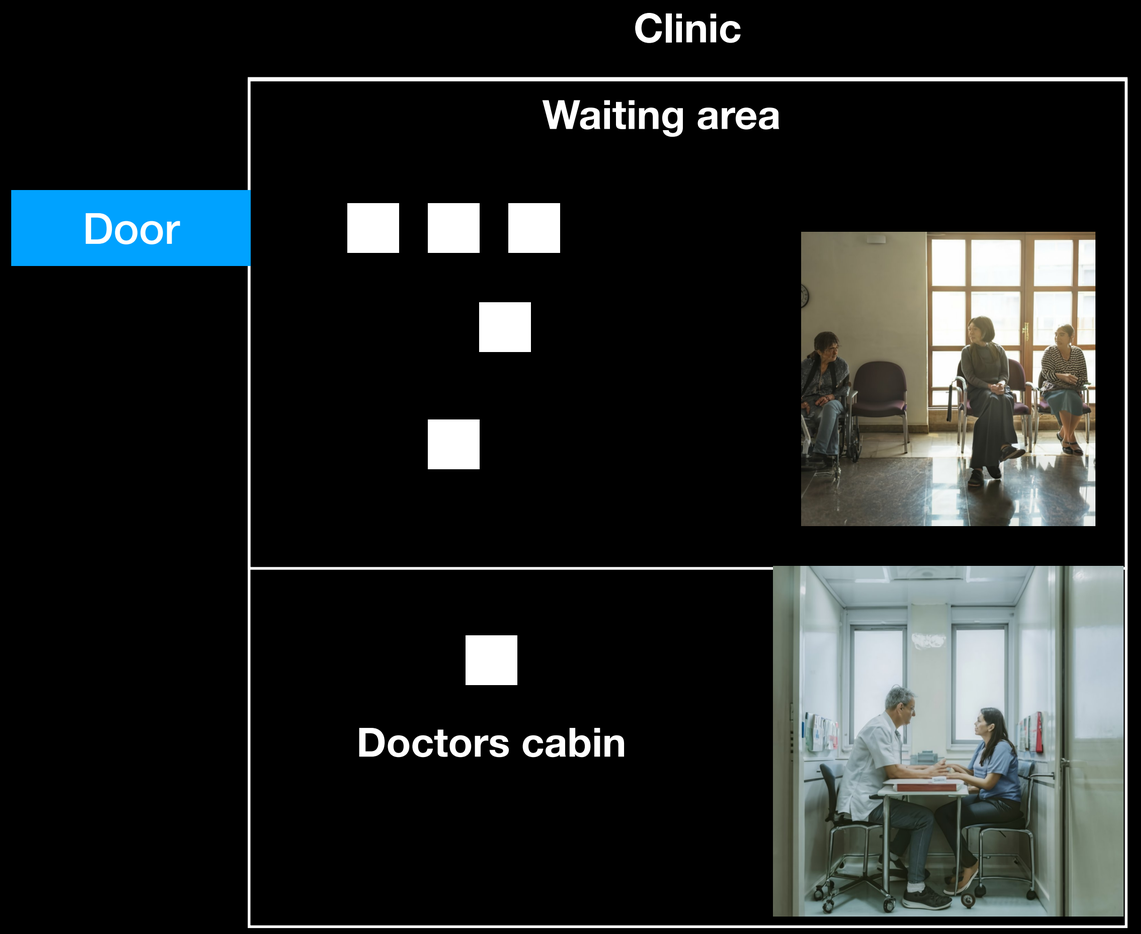
Going to the doctor

You Come back and wait in the waiting room again



Going to the doctor

You Come back and wait in the waiting room again



Your logic is like a doctor

Doctor logic

index.js > ...

```
1  function calculateSum(n) {
2    let ans = 0;
3    for (let i = 1; i<=n; i++) {
4      ans = ans + i;
5    }
6    return ans;
7  }
8
9  let ans = calculateSum(10);
10 console.log(ans);
```

Your logic is like a doctor

Doctor logic

index.js > ...

```
1  function calculateSum(n) {
2    let ans = 0;
3    for (let i = 1; i<=n; i++) {
4      ans = ans + i;
5    }
6    return ans;
7  }
8
9  let ans = calculateSum(10);
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

Your logic is like a doctor

But what if you want to expose this logic to the world?

Doctor logic

index.js > ...

```
1  function calculateSum(n) {
2    let ans = 0;
3    for (let i = 1; i<=n; i++) {
4      ans = ans + i;
5    }
6    return ans;
7  }
8
9  let ans = calculateSum(10);
10 console.log(ans);
```

Your logic is like a doctor

But what if you want to expose this 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

Doctor logic

index.js > ...

```
1  function calculateSum(n) {
2    let ans = 0;
3    for (let i = 1; i<=n; i++) {
4      ans = ans + i;
5    }
6    return ans;
7  }
8
9  let ans = calculateSum(10);
10 console.log(ans);
```

Your logic is like a doctor

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

Ans - By creating an HTTP Server

Doctor logic

index.js > ...

```
1  function calculateSum(n) {
2    let ans = 0;
3    for (let i = 1; i<=n; i++) {
4      ans = ans + i;
5    }
6    return ans;
7  }
8
9  let ans = calculateSum(10);
10 console.log(ans);
```

Your logic is like a doctor

Question - How do I create an HTTP Server?

Ans - Express

Doctor logic

index.js > ...

```
1  function calculateSum(n) {
2    let ans = 0;
3    for (let i = 1; i<=n; i++) {
4      ans = ans + i;
5    }
6    return ans;
7  }
8
9  let ans = calculateSum(10);
10 console.log(ans);
```

Your logic is like a doctor

Question - How do I create an HTTP Server?

Ans - Express

index.js > ...

```
1  function calculateSum(n) {
2    let ans = 0;
3    for (let i = 1; i<=n; i++) {
4      ans = ans + i;
5    }
6    return ans;
7  }
8
9  let ans = calculateSum(10);
10 console.log(ans);
```

index.js

```
1  const express = require("express")
2
3  function calculateSum(n) {
4    let ans = 0;
5    for (let i = 1; i<=n; i++) {
6      ans = ans + i;
7    }
8    return ans;
9  }
10
11 const app = express();
12
13 app.get("/", function(req, res) {
14   const n = req.query.n;
15   const ans = calculateSum(n)
16   res.send(ans);
17 })
18
19 app.listen(3000);
```

Your logic is like a doctor

Question - How do I create an HTTP Server?

Ans - Express

index.js

```
1  const express = require("express")
2
3  function calculateSum(n) {
4    let ans = 0;
5    for (let i = 1; i<=n; i++) {
6      ans = ans + i;
7    }
8    return ans;
9  }
10
11 const app = express();
12
13 app.get("/", function(req, res) {
14   const n = req.query.n;
15   const ans = calculateSum(n)
16   res.send(ans);
17 })
18
19 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

Your logic is like a doctor

Question - How do I create an HTTP Server?

Ans - Express

index.js

```
1  const express = require("express")
2
3  function calculateSum(n) {
4    let ans = 0;
5    for (let i = 1; i<=n; i++) {
6      ans = ans + i;
7    }
8    return ans;
9  }
10
11 const app = express();
12
13 app.get("/", function(req, res) {
14   const n = req.query.n;
15   const ans = calculateSum(n)
16   res.send(ans);
17 })
18
19 app.listen(3000);
```

Deciding the address of the clinic

Your logic is like a doctor

Question - How do I create an HTTP Server?

Ans - Express

Hospital

Doctor 1

index.js > ...

```
1  const express = require("express")
2
3  function calculateSum(a, b) {
4    return a + b;
5  }
6
7  const app = express();
8
9  app.get("/", function(req, res) {
10   const a = req.query.a;
11   const b = req.query.b;
12   const ans = calculateSum(a, b)
13   res.send(ans);
14 })
15
16 app.listen(3001);
```

Doctor 2

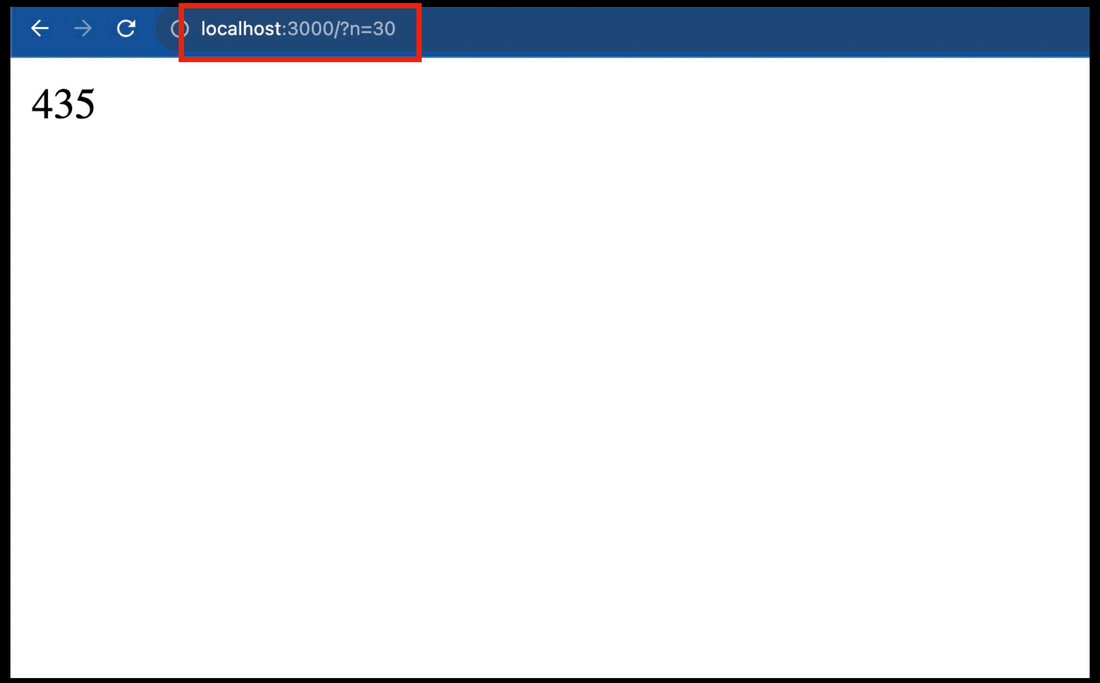
```
1  const express = require("express")
2
3  function calculateSum(n) {
4    let ans = 0;
5    for (let i = 1; i<=n; i++) {
6      ans = ans + i;
7    }
8    return ans;
9  }
10
11 const app = express();
12
13 app.get("/", function(req, res) {
14   const n = req.query.n;
15   const ans = calculateSum(n)
16   res.send(ans);
17 })
18
19 app.listen(3000);
```

Your logic is like a doctor

Question - How do I create an HTTP Server?

Ans - Express

How do patients reach it?



index.js > ...

```
1   const express = require("express")
2
3   function calculateSum(n) {
4     let ans = 0;
5     for (let i = 0; i<n; i++) {
6       ans = ans + i;
7     }
8     return ans;
9   }
10
11  const app = express();
12
13  app.get("/", function(req, res) {
14    const n = req.query.n;
15    const ans = calculateSum(n)
16    res.send(ans.toString());
17  })
18
19  app.listen(3000);
```

OUTPUT DEBUG CONSOLE TERMINAL 1

TERMINAL

```
→ http-server-2 node index.js
```

Your logic is like a doctor

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**

Your logic is like a doctor

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**
5. **403 => you were not allowed in the hospital**

Your logic is like a doctor

Learn by doing, lets create an in memory hospital

You need to create 4 routes (4 things that the hospital can do)

1. GET - User can check how many kidneys they have and their health
2. POST - User can add a new kidney
3. PUT - User can replace a kidney, make it healthy
4. DELETE - User can remove a kidney

Your logic is like a doctor

Learn by doing, lets create an in memory hospital

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]);
```

Your logic is like a doctor

Learn by doing, lets create an in memory hospital

You need to create 4 routes (4 things that the hospital can do)

1. **GET - User can check how many kidneys they have and their health**
2. **POST - User can add a new kidney**
3. **PUT - User can replace a kidney, make it healthy**
4. **DELETE - User can remove a kidney**

index.js > ...

```
1   const express = require("express")
2   const app = express();
3   var users = [{
4       name: 'John',
5       kidneys: [{
6           healthy: false
7       }, {
8           healthy: true
9       }]
10  }]
11
12  app.get("/", function(req, res) {
13                  ?
14  })
15
16  app.post("/", function(req, res) {
17
18  })
19
20  app.put("/", function(req, res) {
21
22  })
23
24  app.delete("/", function(req, res) {
25
26  })
27
28  app.listen(3000);
```

Your logic is like a doctor

Learn by doing, lets create an in memory hospital

1. What should happen if they try to delete when there are no kidneys?
2. What should happen if they try to make a kidney healthy when all are already healthy

Solution - https://gist.github.com/hkirat/7b78356bd28022aecd476d29f3e6645f

index.js > ...

```
1   const express = require("express")
2   const app = express();
3   var users = [{
4       name: 'John',
5       kidneys: [{
6           healthy: false
7       }, {
8           healthy: true
9       }]
10  }]
11
12  app.get("/", function(req, res) {
13                  ?
14  })
15
16  app.post("/", function(req, res) {
17
18  })
19
20  app.put("/", function(req, res) {
21
22  })
23
24  app.delete("/", function(req, res) {
25
26  })
27
28  app.listen(3000);
```

How to test?

POSTMAN