Express.js
CRUD


CRUD
Create
Read
Update
Delete

req.params

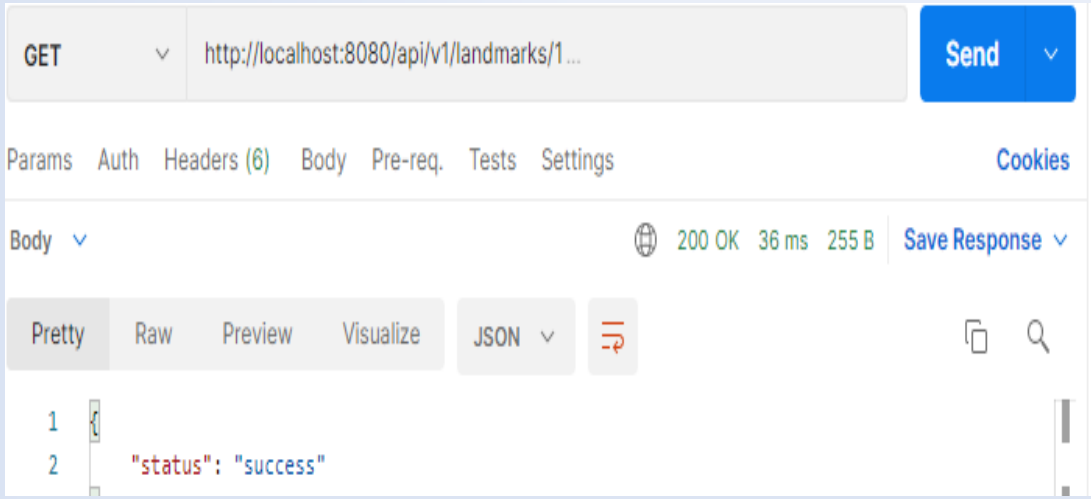
- Lets see a very common behavior we wish our API to have
- Lets say we want to select one landmark by id
  - a parameter is needed to define the id
- **Route parameters ->***named* URL segments, used to capture values specified in position in the URL.
- **req.params object->** used to access the aforesaid values

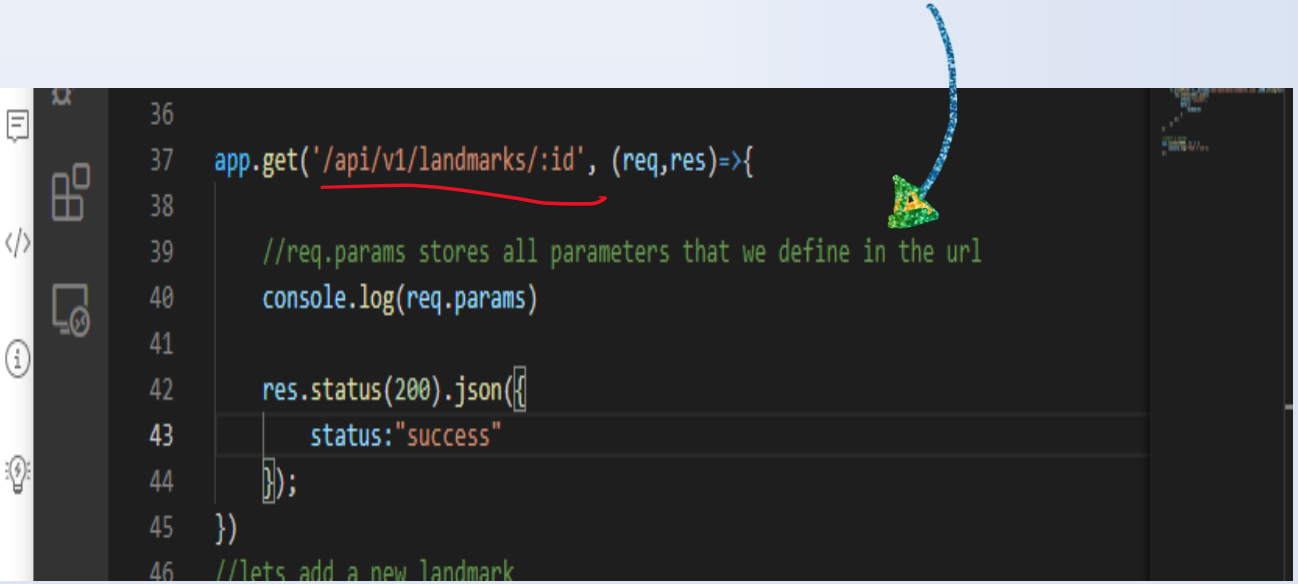
req.params

- Route path: /users/:userId/programmingL/:id
- Request URL: http://localhost:8080/users/2/ programmingL /1
- req.params: { "userId": “2", " id ": “1" }

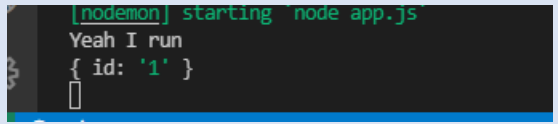
req.params

- Lets see a very common behavior we wish our API to have.
- We want to select one landmark (in this use case scenario) by id



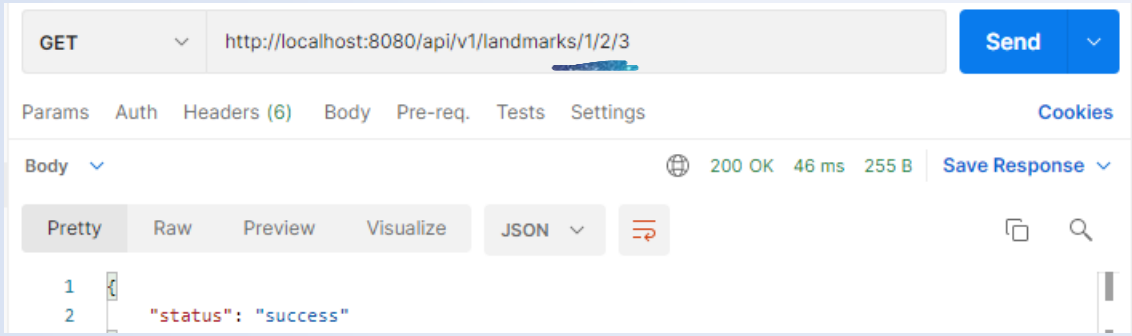


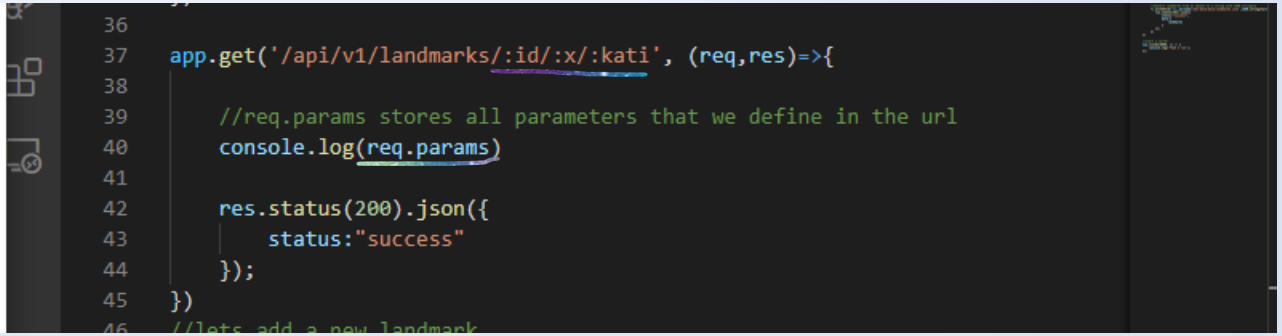
here is our param

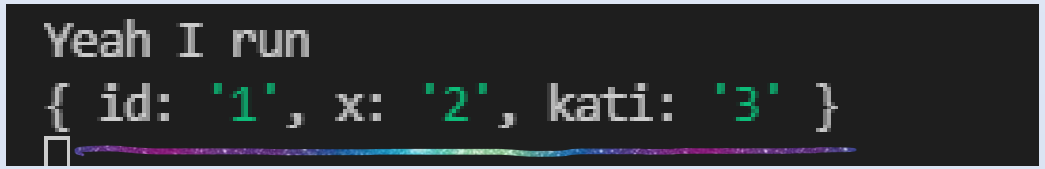


req.params

- Could we define more than one parameters in the url?
- Yeap we can!



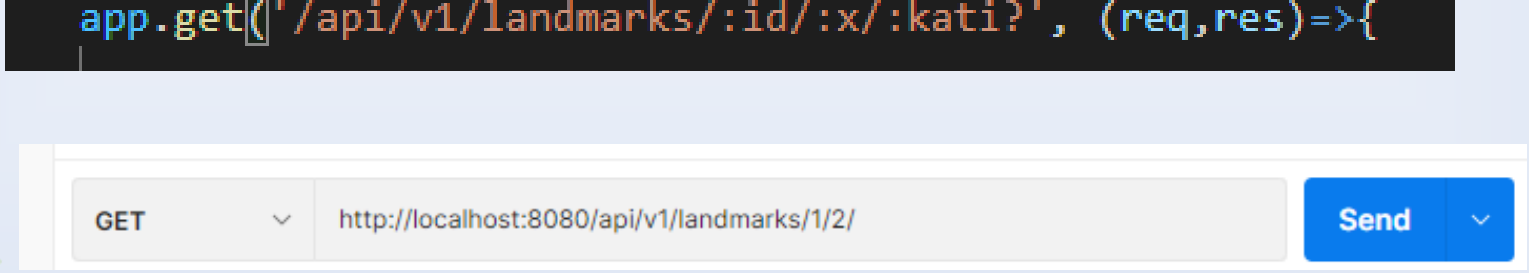




req.params

- We can also define optional parameters
- That we may or may not specify at the endpoint

```
app.get('/api/v1/landmarks/:id/:x/:kati?', (req,res)=>{
```




```
Yeah I run
{ id: '1', x: '2', kati: undefined }
```

Select by id

- We want to find a landmark by its id
- So how could we do that?

Select by id

- We want to find a landmark by its id
- So how could we do that?

- We have a landmarks object… so we can use the *find* method

- find() method -> returns the value of the **first element** in the provided array that satisfies the provided testing function
- no values -> undefined is returned.

Select by id

find() method -> returns the value of the **first** element in the provided array that satisfies the provided testing function

i.E

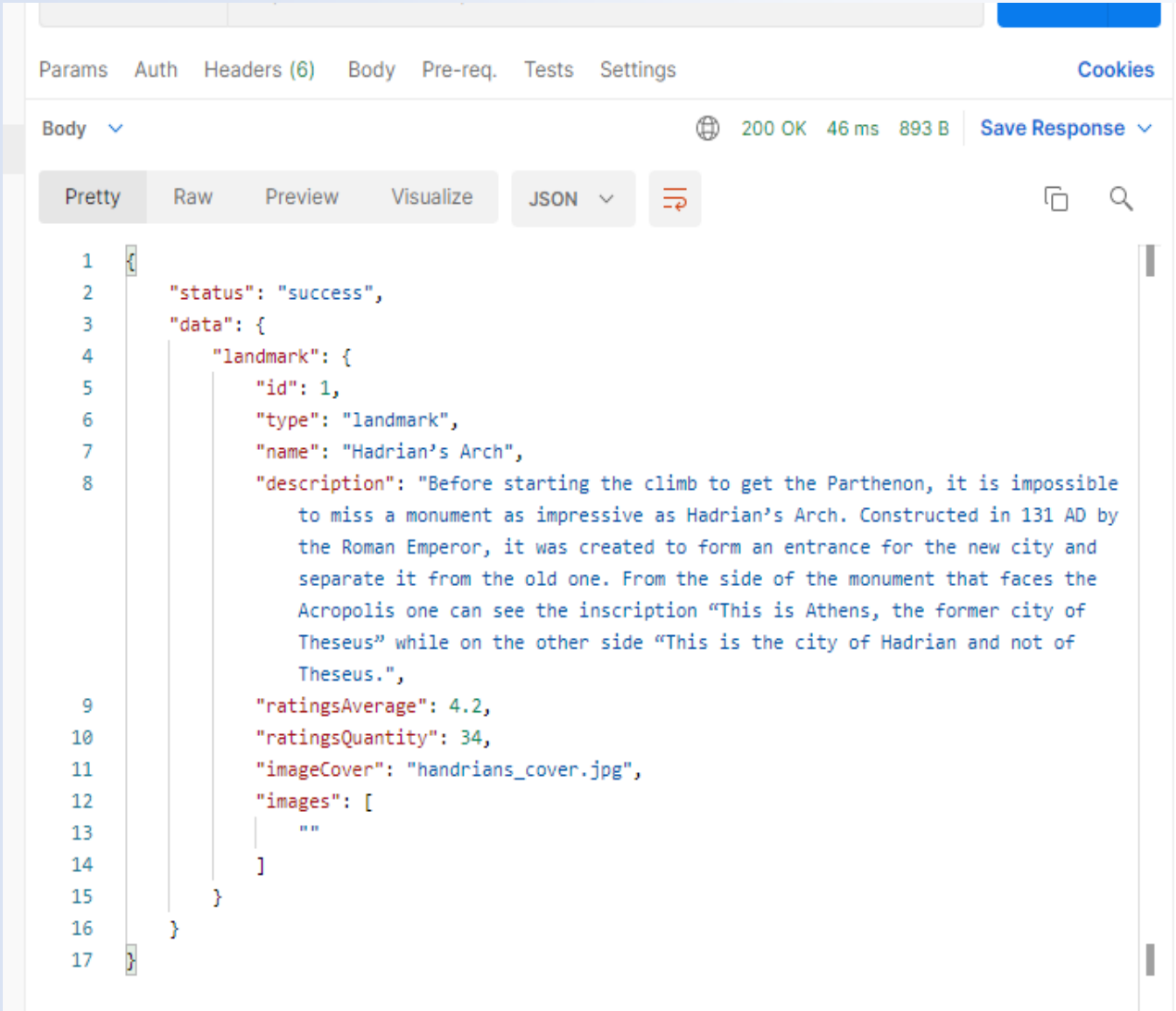
```
var array1 = [2, 12, 1, 3, 13, 4, 66];

var found = array1.find(element => element > 10);

console.log(found);// expected output: 12
```

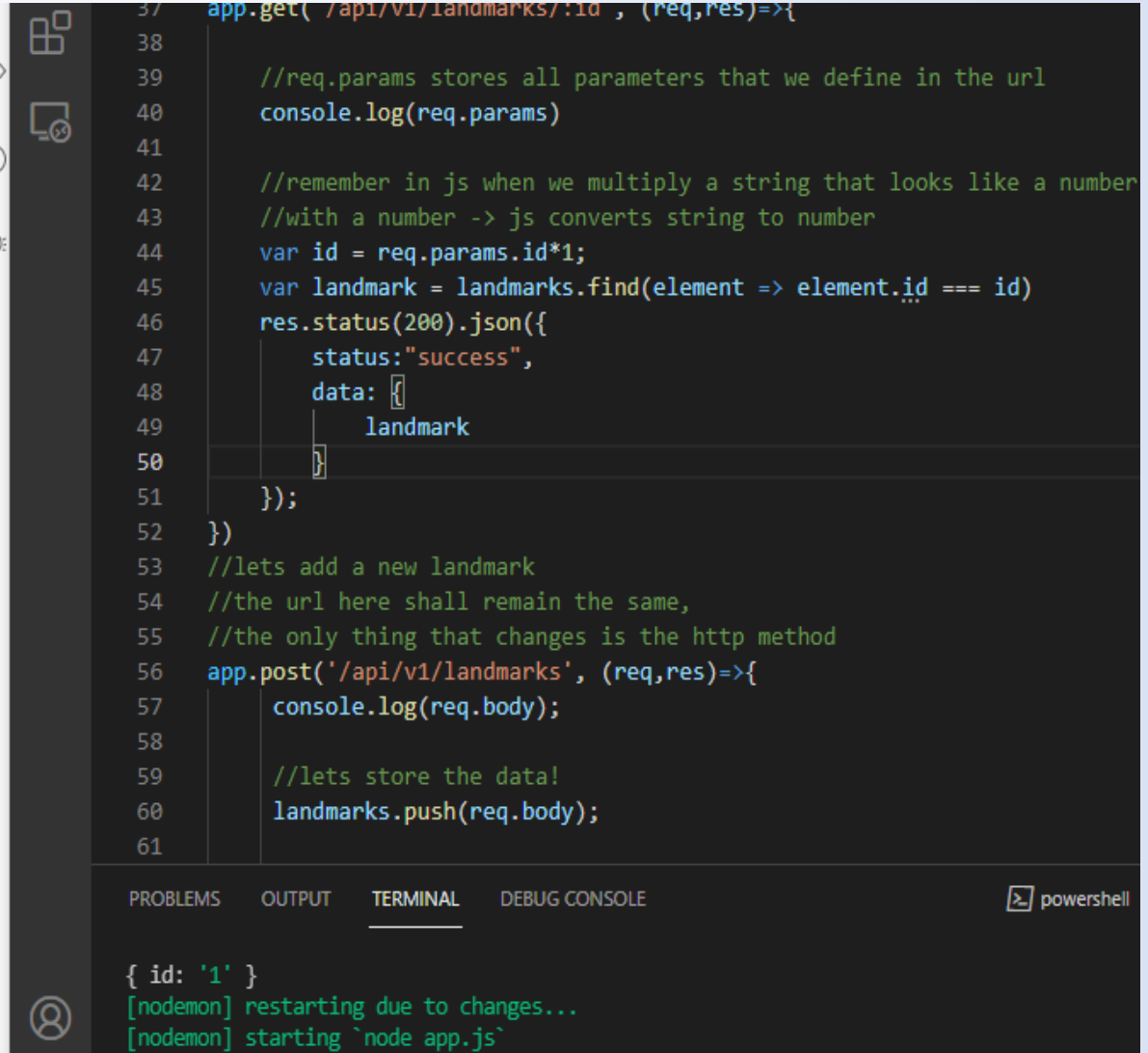
Select by id

Check here: id is string in the request so we need to convert it to number!

```
//remember in js when we multiply a string that looks like a numbe
//with a number -> js converts string to number
var id = req.params.id*1;
var landmark = landmarks.find(element => element.id === id)
```

Select by id


Params
Auth
Headers (6)
Body
Pre-req.
Tests
Settings
Cookies
Body
200 OK
46 ms
893 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status": "success",
"data": {
"landmark": {
"id": 1,
"type": "landmark",
"name": "Hadrian's Arch",
"description": "Before starting the climb to get the Parthenon, it is impossible to miss a monument as impressive as Hadrian's Arch. Constructed in 131 AD by the Roman Emperor, it was created to form an entrance for the new city and separate it from the old one. From the side of the monument that faces the Acropolis one can see the inscription "This is Athens, the former city of Theseus" while on the other side "This is the city of Hadrian and not of Theseus.",
"ratingsAverage": 4.2,
"ratingsQuantity": 34,
"imageCover": "handrians_cover.jpg",
"images": [
""
]
}
}
}


app.get('/api/v1/landmarks/:id', (req,res)=>{
//req.params stores all parameters that we define in the url
console.log(req.params)
//remember in js when we multiply a string that looks like a number
//with a number -> js converts string to number
var id = req.params.id*1;
var landmark = landmarks.find(element => element.id === id)
res.status(200).json({
status:"success",
data: {
landmark
}
});
})
//lets add a new landmark
//the url here shall remain the same,
//the only thing that changes is the http method
app.post('/api/v1/landmarks', (req,res)=>{
console.log(req.body);
//lets store the data!
landmarks.push(req.body);
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
powershell
{ id: '1' }
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`

Select by id

Handle the situation of no such id in Json

```
    //remember in js when we multiply a string that looks like a number
    //with a number -> js converts string to number
    var id = req.params.id*1;
    var landmark = landmarks.find(element => element.id === id)
    if(!landmark){
        res.status(404).json({
            status:"fail",
            message: "Not Found"
        });
    }
    res.status(200).json({
        status:"success",
        data: {
            landmark
        }
    });

})
```

Multiple callback functions

```
app.get('/api/v1/landmarks',(req,res,next)=>{
    console.log("More than one callback can handle a route, dont' forget next!");
    next();
},(req,res)=>{
    res.status(200).json({
        status:"success",
        results: landmarks.length,
        data:{
            landmarks
        }
    });
})
```

More than one callback function can handle a route-> don’t forget to specify next

Array of route handlers

```
var a=(req,res,next)=>{
    console.log("hi there");
    next();
}
var b=(req,res,next)=>{
    console.log("My friend");
    next();
};
var c=(req,res,)=>{
    res.send('Hello!')
};

app.get('/api/v1/array',[a,b,c]);
```

Response methods

- *methods* on the response object (**res**) in the following table can **send** a response to the client, and **terminate** the request-response cycle.

- Note that If none of these methods are called from a route handler-> the client request will be left **hanging**

Method	Description
res.download()	Prompt a file to be downloaded.
res.end()	End the response process.
res.json()	Send a JSON response.
res.jsonp()	Send a JSON response with JSONP support.
res.redirect()	Redirect a request.
res.render()	Render a view template.
res.send()	Send a response of various types.
res.sendFile()	Send a file as an octet stream.
res.sendStatus()	Set the response status code and send its string representation as the response body.

Source https://expressjs.com/

Update Data

- For updating data we have two methods:
- PUT ->is used to modify an existing entity
  - it replaces an entity.-> If we don’t include a property that an entity contains, it should be removed
- PATCH -> is used to apply a **partial** modification to a resource.
  - Used to update only the properties we wish
- We shall see such an example when we **get to databases**

Delete

```
//we shall see more in databases about that
app.delete('/api/v1/landmarks/:id', (req,res)=>{
    //204 status usually stands for no content
    //this is what we usually use with delete
    res.status(204).json({
        status:"success",
        data: null
    });

})
//start a server
```

Structure our code a little bit better

- Now, in order to be a little bit more organized, we are going to separate the **http methods** from the **route handler functions**

Structure our code a little bit better

```
const getAllLandmarks = (req,res)=>{

    res.status(200).json({
        status:"success",
        results: landmarks.length,
        data:{
            landmarks
        }
    });
}

const getLandmarkById = (req,res)=>{

    //req.params stores all parameters that we define in t
    console.log(req.params)

    //remember in js when we multiply a string that looks
    //with a number -> js converts string to number
    var id = req.params.id*1;
    var landmark = landmarks.find(element => element.id ==
    if(!landmark){
        res.status(404).json({
            status:"fail",
            message: "Not Found"
        });
    }
```

```
app.get('/api/v1/landmarks',getAllLandmarks)

app.get('/api/v1/landmarks/:id',getLandmarkById )

//lets add a new landmark
app.post('/api/v1/landmarks',addLandmark)

//update one property with patch
app.patch('/api/v1/landmarks/:id', updateLandmarkById)

//we shall see more in databases about that
app.delete('/api/v1/landmarks/:id', deleteLandmarkById )

//start a server
app.listen(8080, () => {
    console.log('Yeah I run');
});
```

assign callback functions(route handlers) to variables
(req,res)=>{} this callback funtion is called the route handler

Can we make it better?

- **app.route()** method -> **returns instance of a single route**, which we can then use to handle HTTP methods
- This allows us to group all same urls… avoid duplicate route names !

- Lets see how

Can we make it better?

```
app.route('/api/v1/landmarks')
.get(getAllLandmarks)
.post(addLandmark)


app.route('/api/v1/landmarks/:id')
.get(getLandmarkById)
.patch(updateLandmarkById)
.delete(deleteLandmarkById)
```

Sure!

To be continued…

https://expressjs.com/en/resources/middleware
https://www.geeksforgeeks.org/express-js-app-route-function/