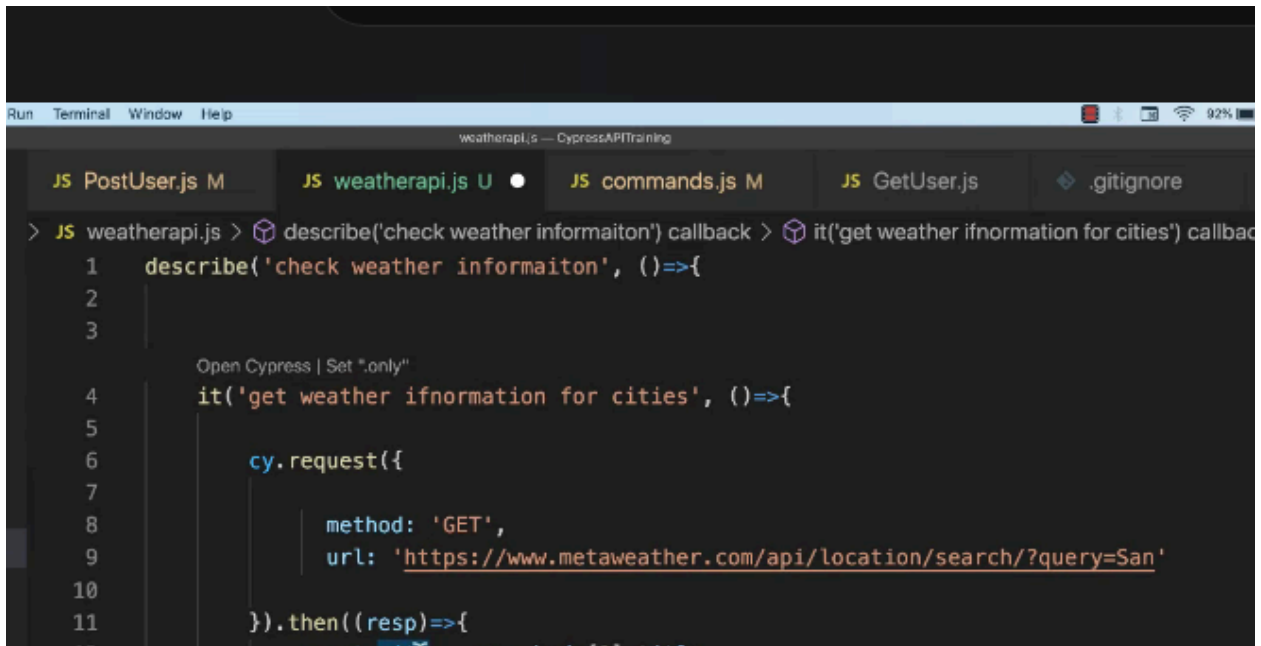


# How to pass single value or complete array from one request to another request using chaining in Cypress

## Chaining

A screenshot of a code editor window titled 'weatherapi.js - CypressAPITraining'. The editor shows a JavaScript file named 'weatherapi.js' with the following code:

```
1 describe('check weather informaiton', ()=>{
2
3
4   it('get weather ifnformation for cities', ()=>{
5
6     cy.request({
7
8       method: 'GET',
9       url: 'https://www.metaweather.com/api/location/search/?query=San'
10
11     }).then((resp)=>{
12       const title = resp.body[0].title
```

The code demonstrates how to chain a Cypress request to a 'then' callback to pass data to the next step. The editor interface includes a top bar with 'Run', 'Terminal', 'Window', and 'Help' menus, and a tab bar with files like 'JS PostUser.js', 'JS weatherapi.js', 'JS commands.js', 'JS GetUser.js', and '.gitignore'.

Using return also we can pass data to next **then** call.

Supply the same value which you are returning in the corresponding **then** statement to work with that variable.

```

9      url: 'https://www.metaweather.com/api/location/search/?query=San'
10
11    }).then((resp)=>{
12      const city = resp.body[0].title
13      return city
14    })
15    .then((city)=>{
16      cy.request({
17        method: 'GET',
18        url: 'https://www.metaweather.com/api/location/search/?query='+city
19      })

```

```

17      url: 'https://www.metaweather.com/api/location/search/?query='+city
18    }).then((resp)=>{
19      expect(resp.status).to.eq(200)
20      expect(resp.body[0]).to.have.property('title', city)
21    })
22
23  })

```

Another chaining

```

27
28
29  Open Cypress | Clear ".only"
30  it.only('get weather information for all cities', ()=>{
31    //1st request: GET locations
32    cy.request({
33      method: 'GET',
34      url: 'https://www.metaweather.com/api/location/search/?query=Am'

```

```

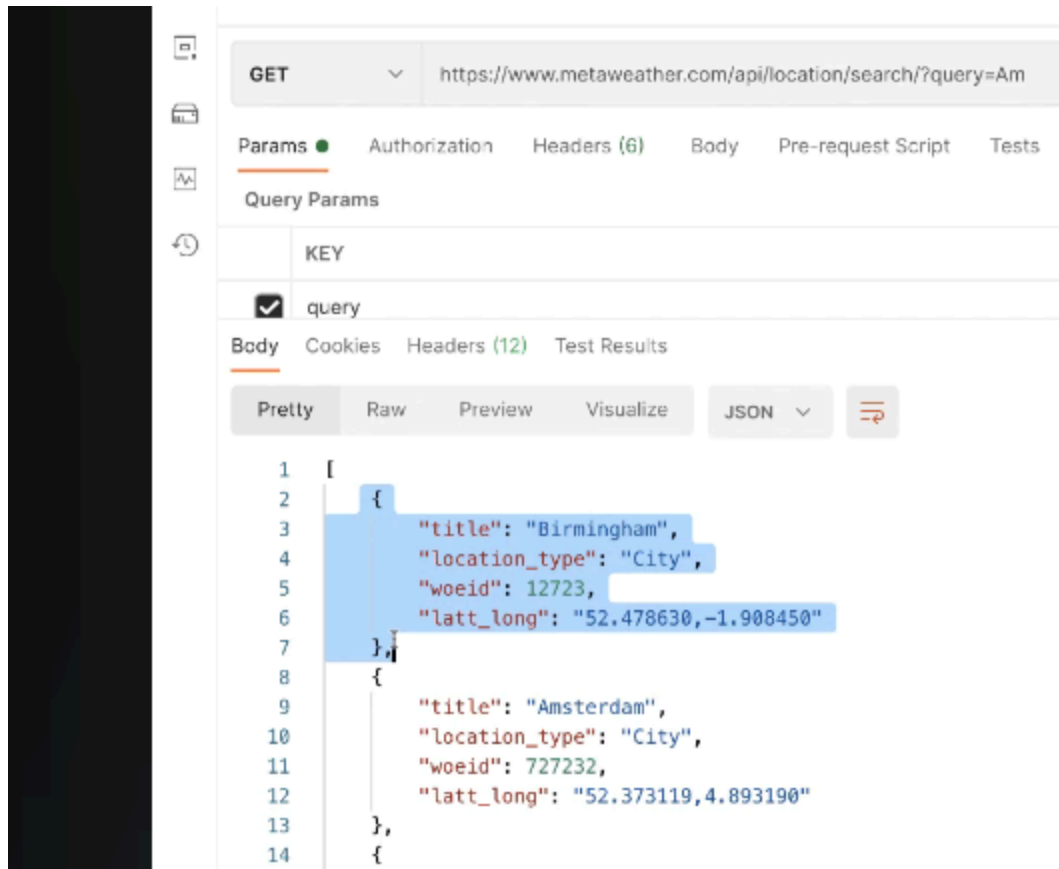
35    }).then((resp)=>{
36      const location = resp.body
37      return location
38    })

```

Using for loop to iterate over every location name.

Inside the for loop itself the request method will keep calling GET method for different locations.

Sample output:



The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `https://www.metaweather.com/api/location/search/?query=Am`
- Params:** query (checked)
- Body:** Pretty (selected), Raw, Preview, Visualize, JSON (dropdown), and a refresh icon.
- Response Body (JSON):**

```
1 [
2   {
3     "title": "Birmingham",
4     "location_type": "City",
5     "woeid": 12723,
6     "latt_long": "52.478630,-1.908450"
7   },
8   {
9     "title": "Amsterdam",
10    "location_type": "City",
11    "woeid": 727232,
12    "latt_long": "52.373119,4.893190"
13  },
14  {
```

```
36 const location = resp.body
37 return location
38 })
39 .then((location)=>{
40
41   for(let i=0; i< location.length; i++){
42     //2nd request for the first location/city
43     cy.request({
44       method: 'GET',
45       url: 'https://www.metaweather.com/api/location/search/?query='+location[i].title
46     }).then((resp)=>{
47       expect(resp.status).to.eq(200)
48       expect(resp.body[0]).to.have.property('title', location[i].title)
49     })
50   }
51 }
52
53 })
```