

Http calls and Exception handling

Animation in Javascript

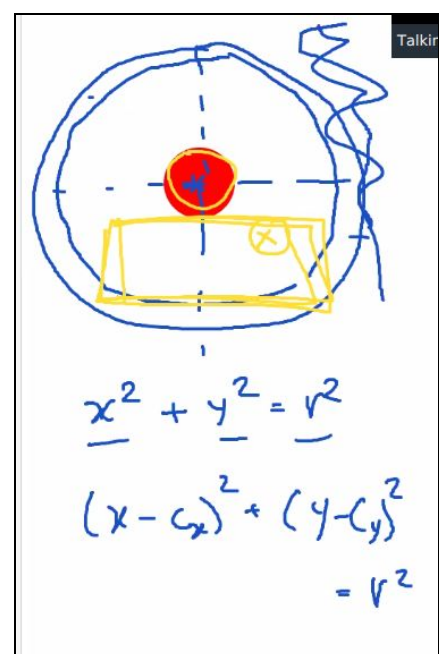
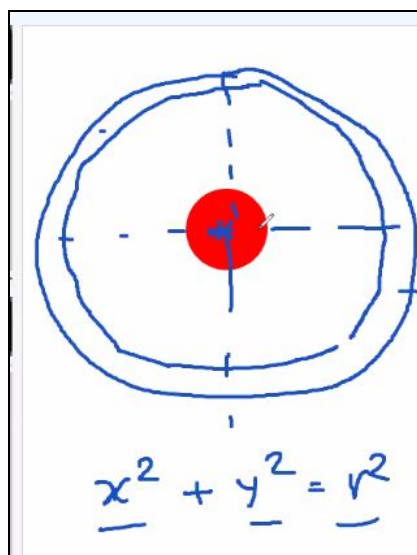
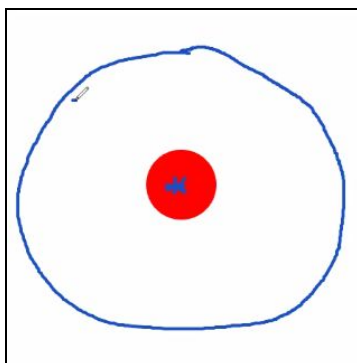
```
HTML
1 <div id="c1">
2 </div>
```

```
#c1 {
  height: 50px;
  width: 50px;
  background-color: red;
  position: absolute;
  border-radius: 50%;
}
```

Inline style

```
="top: 100px; left: 100px;">
```

Giving ball a circular path, moving in circular direction



Animation is very difficult in javascript you should not do it unless you need it.
Javascript shines in interactivity. Adding animation using javascript is hard that's why external libraries are used.

What can we do using javascript

1. Add interactivity with user/client.
2. Add animations.
3. Perform network request.

Today's class is dedicated to this Network request and http calls.

Http calls

Fetching data from a server

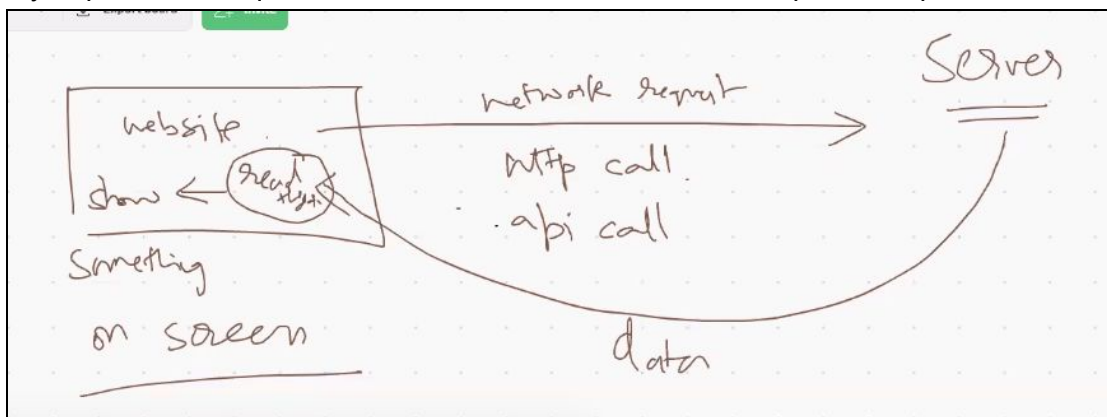
- You provide a link and hit enter. At this moment http sends a request to the server whose address you have given in the url and then server gives a response called api and then javascript reads this api and shows you the data.



- Whenever you click on only of these links below:

| | | | | | | | |
|--|-----------------------|----|---|----|-----|-----|-------|
| ▶ Java Script 102 - Post Class | Oct 16, 2020 11:59 PM | 0 | 0 | 0 | 112 | 99 | 0/211 |
| ▶ Weekly Assignment | Oct 16, 2020 11:59 PM | 0 | 0 | 0 | 164 | 47 | 0/211 |
| ▶ JS Assignment - Post Class | Oct 15, 2020 11:59 PM | 63 | 0 | 61 | 88 | 60 | 0/211 |
| ▶ BOM and DOM with Javascript | Oct 15, 2020 11:59 PM | 55 | 0 | 55 | 49 | 107 | 0/211 |

- Some progress bar comes and then the numbers and other things are loaded. Progress is indicating that the data is coming from the server which takes some time.
- Internally http sends a request to a server. This is also called as http call or api call.



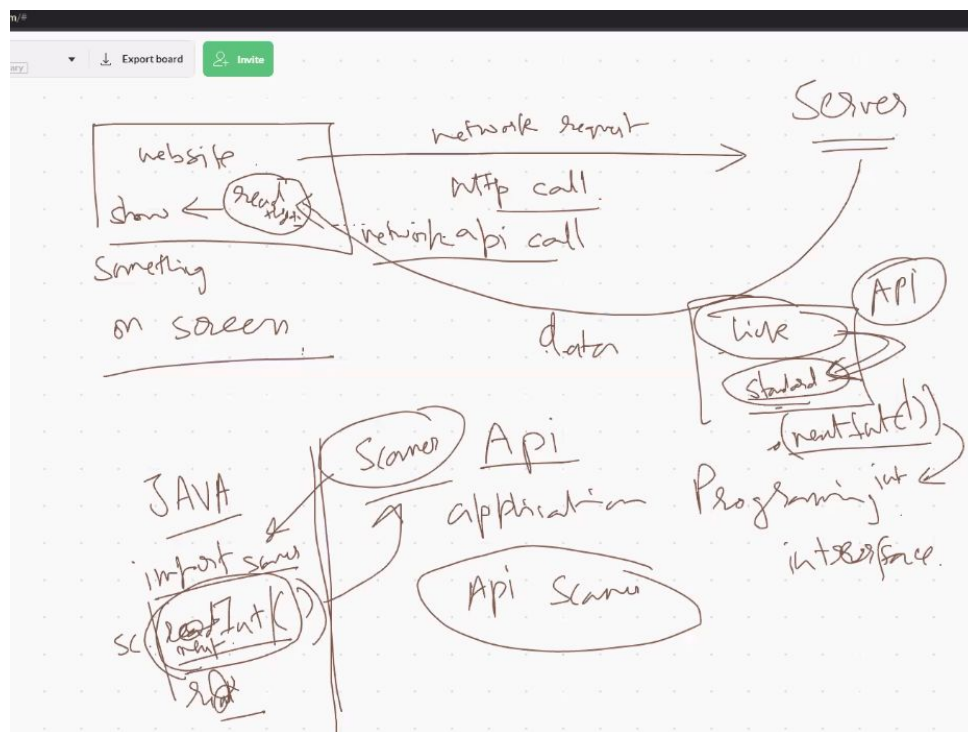
- Then server gives response in the form of api. This data is read using javascript and then displayed by using combination of javascript, html and css.

What is an api?

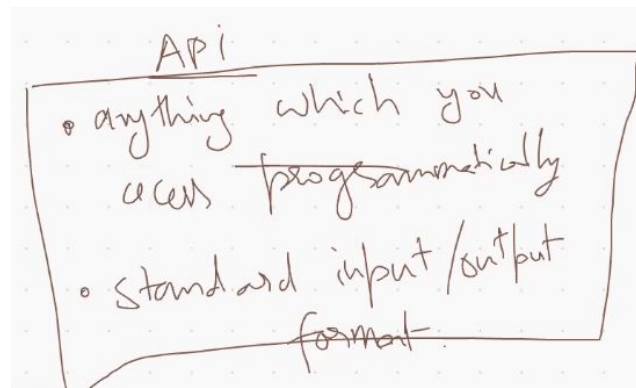
- API: Application Programming Interface.
- Its an interface between 2 programming systems.
- Any function or operation in programming who has a standard(one that doesn't change) input and output. That function/operation is an api.
- E.g Scanner api documentation.
 - `sc.nextInt()` can also be called as scanner api call. Because it has a standard input always i.e `sc.nextInt()` will take input from user and standard output it will return user's input to cpu.
- For differentiation purposes, google.com is not an api, why?, because the link google.com is standard link that means input never changes but the response always changing, sometimes logo is changed sometimes a different animation,, different adds so output is not standard.

Network api

Api have one link we react to that link and it will always return the required stuff in a standard format.



- A function is an api call
- Network request is an api call
- Api: anything which u access programmatically(using code) and it has a standard input and output format.



Example of api link

<https://jsonplaceholder.typicode.com/users>

Example of api response

```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-
      bs": "harness real-time e-markets"
    }
  }
]
```

- The link is returning this, it's an array structure which is simply returning to you standard key value pairs.
- All the api which we get on internet is a rest api.
- REST stands for Representational State transfer. A REST API is a way for two computer systems to communicate over HTTP in a similar way to web browsers and servers.

This a url for network request. Network request means requesting some information from other server.

<https://jsonplaceholder.typicode.com/users>

This is the data that the server will send back which is called as response

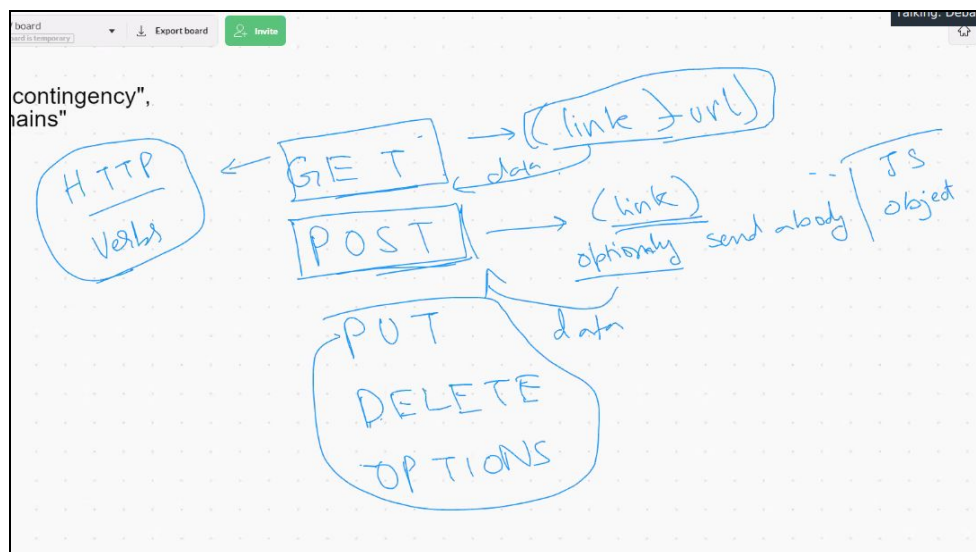
```
[  
  {  
    "id": 1,  
    "name": "Leanne Graham",  
    "username": "Bret",  
    "email": "Sincere@april.biz",  
    "address": {  
      "street": "P.O. Box 821, 81234",  
      "suite": "Apt. 565",  
      "city": "Newburgh",  
      "state": "New York",  
      "zip": "12550-3717",  
      "country": "USA"  
    }  
  }  
]
```

data
return data

- Then we read this data(response).
- Then we show the data on our website that we got.

Topic: how can we make network request

- There are different request methods using which we can make a network request.
 - a. GET
 - b. POST
 - c. PUT
 - d. DELETE
 - e. OPTIONS
- GET, POST, these http verbs.
- Get requests are only made using just a link and it returns data. Its generally used for fetching data
- Post request also needs a link and optionally POST can also send the body to the server and server returns the data.



- **Get takes link and return data**
- **Post takes link and optionally takes the body with data(to send the data to the server) and it can return data**

- These are called Request Methods.
- For exploring, go to json placeholder typicode(<https://jsonplaceholder.typicode.com/>), Go to routes section, GET request, Post request, Go to the guide.

How do we get data using javascript and show it(log) to the console

```
<script>
  function getData() {
  }
</script>
</head>
<body onLoad="loadFunction()">
  <button onClick="getData()">Get Data</button>
  <div id="d1"></div>
</body>
```

- In general get is used for getting data from server and post is used for sending data to the servers
- Xhr request: Oldest way to get data to from server. Almost no one uses it. Only old systems of Banks may using it.

```
document.getElementById(" ")
```

This is a javascript api

Modern way of getting data from the server

- “fetch” api.

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
  .then((response) => response.json())
  .then((json) => console.log(json))
```

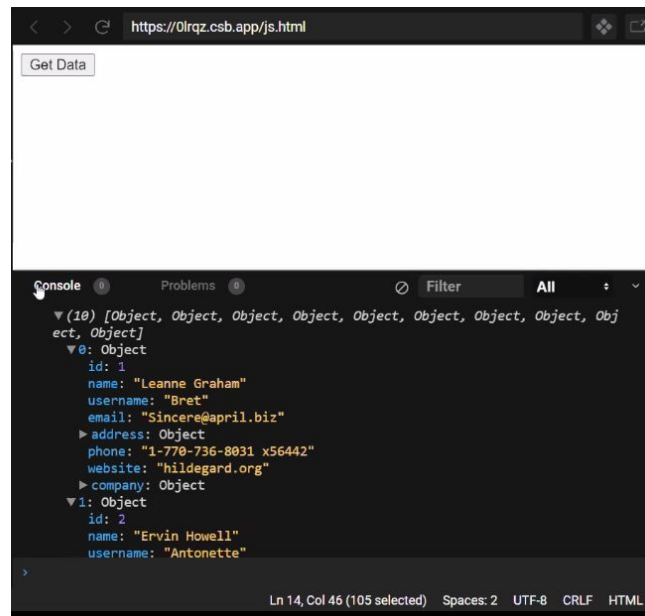
Use of fetch

```
const url = "https://jsonplaceholder.typicode.com/users"
```

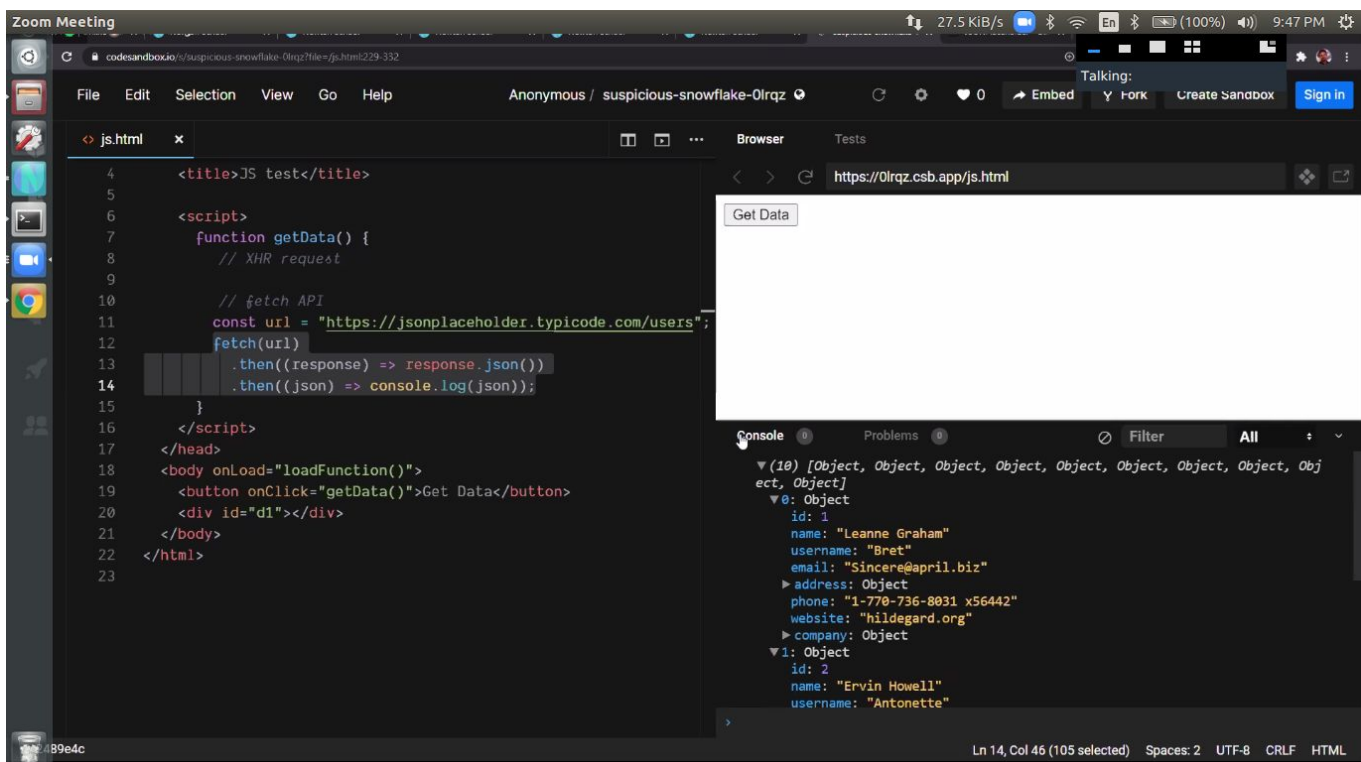
storing string representation of url in 'url' name variable.

```
// fetch API
const url = "https://jsonplaceholder.typicode.com/users";
fetch(url)
  .then((response) => response.json())
  .then((json) => console.log(json));
</script>
```

Example for fetching a request and showing it console.



After clicking on 'Get Data' you got the response from (https://jsonplaceholder.typicode.com//)



Full screen shot of the example for fetching response and showing to console.

How data comes and how js evaluate it

- `fetch(url)` -> url is the full link `fetch(url)` will fetch the data from this link
- `.then()` do this
- `.then()` do this
- `fetch(url).then(do something)`
- 'then' is a function which will take another function as an arg, then we execute that function

```
fetch(url).then(function() {  
  console.log("Exceuted at then");  
})
```

```
// fetch API - GET  
const url = "https://jsonplaceholder.typicode.com/users";  
fetch(url).then(function () {  
  console.log("Exceuted at then");  
});  
}
```

- When function doesn't have any name it is called as anonymous function. Usually these are defined at the exact same place where it will be used. Like in the example above the function is defined inside 'then' and it doesn't have any name.

You can also do it this way

```
fetch(url).then(tobeExecutedInThen);  
}  
  
function tobeExecutedInThen() {  
  console.log("Exceuted at then");  
}
```

If we declare function outside we have to give a name (called as identifier)

Reminding that functions can be used as variables. They are first class members in javascript. Here 'then' function is using another function 'tobeExecutedInThen()' as an argument.

Another Example of the same thing

```
const tobeExceuted = function () {  
  console.log("Exceuted at then");  
};
```

Anonymous function is defined and stored in a variable 'tobeExceuted'
You can see it in way that now the name of the anonymous function is tobeExceuted. You can access it using its name.

```
fetch(url).then(tobeExceuted);
```

Accessed that function

Full picture of above two

```
const url = "https://jsonplaceholder.typi  
fetch(url).then(tobeExceuted);  
}  
  
const tobeExceuted = function () {  
  console.log("Exceuted at then");  
};
```

only the part selected is the anonymous function

- Function is assigned to a variable in a way the function name is now this variable. So that variable is used in then(here)

```
fetch(url).then(tobeExceuted);  
}  
  
const tobeExceuted = function () {  
  console.log("Exceuted at then");  
};  
  
function tobeExceuted() {  
  console.log("Exceuted at then");  
}
```

The two function declared in this picture are one in the same thing But which will be called is discussed below

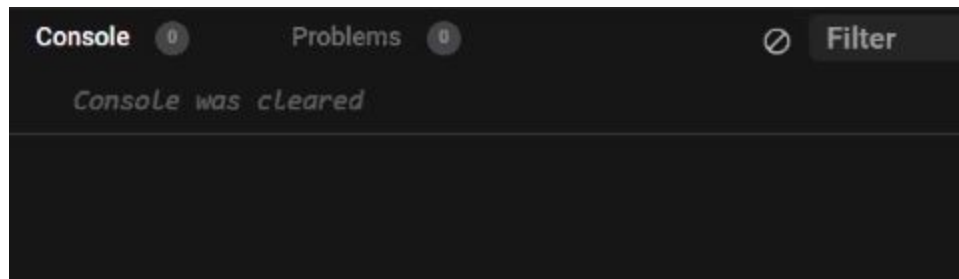
```

const url = "https://jsonplaceholder.typicode.com/todos/1";
fetch(url).then(tobeExceuted);
}

const tobeExceuted = function () {
  console.log("Exceuted at then 1");
};

function tobeExceuted() {
  console.log("Exceuted at then 2");
}
</script>

```



Nothing has happened

- This tells that javascript is not very good about telling the error. If a complicated ide is used that would have shown error.
- So don't redeclare the same thing

Some Points to remember:

- ECMAScript (ES5, ES6) these are the versions of javascript.
- Best practice is to give semicolon when you use a variable.
- Best practice of using the const.

Next

How do we get back the data? How can we use the data received from the server? Where it is received?

```

const tobeExceuted = function (resp) {
  console.log(resp, "Executed", "with resp");
};

```

'resp' named argument is passed in the function

```

fetch(url).then(tobeExceuted);

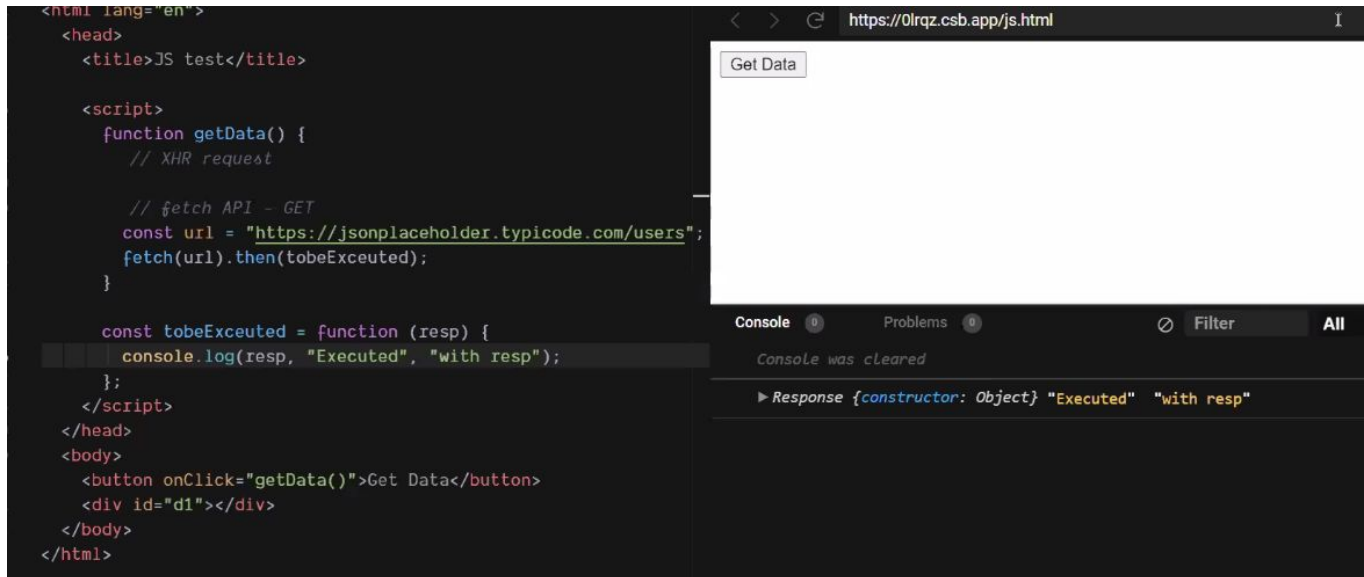
```

'tobeExceuted' is passed as an argument in 'then' function.

- 'fetch' function will fetch the response from the url, and pass the response to the argument of 'then' function. Which means the response fetched by 'fetch' function is passed to the 'tobeExceuted' function.

- This response can now be accessed using 'resp' variable.

Code + browser output + console



Observe console: It has received the response.

Comma separated things in 'console.log()' function is replaced by space separated things.

Explanation in detail.

- Then is always passing the response of previous function.
- For example in the below example, function returns integer 123. If anywhere 'tobeExceuted' function is called it will return this integer (123).
- If this function is called inside 'then' function it will return 123 to next then function.

```
const tobeExceuted = function (resp) {
  console.log(resp, "Executed", "with resp");
  return 123;
};
</script>
```

- This is called as chaining. 'then' is a chainable function.

```
fetch(url)
  .then(tobeExceuted)
  .then(function (r) {
    console.log(r);
  });
}
```

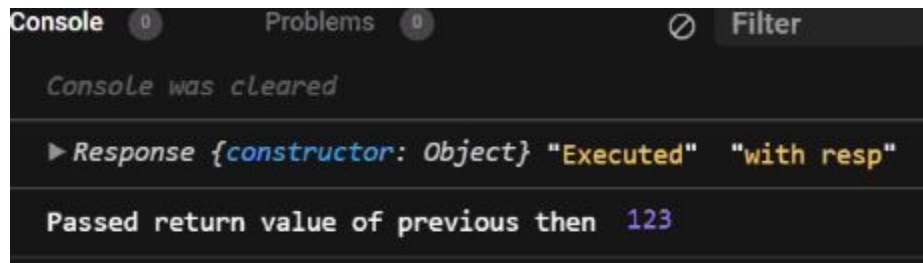
```
Console was cleared

▶ Response {constructor: Object} "Executed" "with resp"

123
```

Even more clear

```
fetch(url)
  .then(tobeExceuted)
  .then(function (r) {
    console.log("Passed return value of previous then", r);
  });
}
```



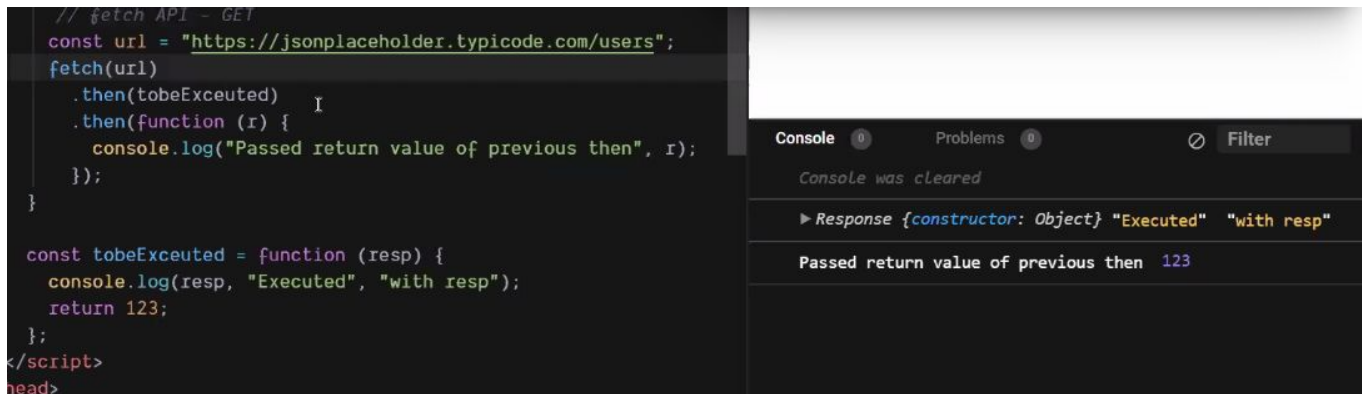
Console 0 Problems 0 Filter

Console was cleared

► Response {constructor: Object} "Executed" "with resp"

Passed return value of previous then 123

Full Image for clarity.




```
// fetch API - GET
const url = "https://jsonplaceholder.typicode.com/users";
fetch(url)
  .then(tobeExceuted)
  .then(function (r) {
    console.log("Passed return value of previous then", r);
  });

const tobeExceuted = function (resp) {
  console.log(resp, "Executed", "with resp");
  return 123;
};
</script>
<head>
```

- Its telling that
 - tobeExceuted is returning 123 so this 123 is passed to next then in the variable called r.

Points to remember:

- Fetch is a part of classes called promises.
- You can do as many .then as many you want
- 'then' takes a function and executes it and the response of this function is passed to the next 'then'.
-  This symbol in codesandbox console clears the console.

Promises:

- Promise is something which takes a function, executes it with the response of the previous promise or then returns a new promise, or data.
- The below image is showing that fetch function is a promise.

```

fetch(url) // Promises
  .then(tobeExceuted)
    // takes a function, executes it with
    // the response of the previous promise or then
    // returns a new promise, or data
  .then(function (r) {
    console.log("Passed return value of previous then", r);
  });

```

Just read the comments by debanshu sir.

- The image below shows that we are getting response from the url.
- This response is passed to 'tobeExecuted'
- 'tobeExecuted' is returning 123.
- 123 is passed to next 'then' and 'r' is receiving 123.
- Console is displaying the 'r'.

```

8 // xhr request
9
10 // fetch API - GET
11 const url = "https://jsonplaceholder.typicode.com/users";
12 fetch(url) // Promises
13   .then(tobeExceuted)
14     // takes a function, executes it with
15     // the return value of the previous promise or then
16     // returns a new promise, or data
17   .then(function (r) {
18     console.log("Passed return value of previous then", r);
19   });
20 }
21
22 const tobeExceuted = function (resp) {
23   console.log(resp, "Executed", "with resp");
24   return 123;
25 };

```

- Writing input parameter(sometimes called argument) is imp otherwise we could not used the return value from the previous function.
- undefined will be printed because 1st anonymous function is not returning anything for the last 'then'.

```
// fetch API - GET
const url = "https://jsonplaceholder.typicode.com/users";
fetch(url) // Promises
  .then(tobeExceuted)
  // takes a function, executes it with
  // the return value of the previous promise or then
  // returns a new promise, or data
  .then(function (r) {
    console.log("Passed return value of previous then", r);
  })
  .then(function (r) {
    console.log("some data", r); // ? what will it print
  });
}
```

To: Everyone

Console Problems Filter All

Console was cleared

► Response {constructor: Object} "Executed" "with resp"

Passed return value of previous then 123

some data undefined

Observe undefined is printed in console

Now observe when we returned 'kuch toh hoga' in 1st anonymous function, the 'r' parameter in 2nd anonymous function received it. And then printed in the console.

```
// fetch API - GET
const url = "https://jsonplaceholder.typicode.com/users";
fetch(url) // Promises
  .then(tobeExceuted)
  // takes a function, executes it with
  // the return value of the previous promise or then
  // returns a new promise, or data
  .then(function (r) {
    console.log("Passed return value of previous then", r);
    return "kuch toh hoga";
  })
  .then(function (r) {
    console.log("some data", r); // ? what will it print
  });
}
```

To: Everyone

Console Problems Filter

Console was cleared

► Response {constructor: Object} "Executed" "with resp"

Passed return value of previous then 123

some data kuch toh hoga

Arrow function or Lambda function

```
const tobeExceuted = (resp) => {  
  console.log(resp, "Executed", "with resp");  
  return 123;  
};  
/script>
```

- In place of function keyword. You can use (combination of equal to and greater than sign) =>. Also called as arrow operator.
- 'function(resp)' is replaced by '(resp)=>' other things remains the same.

```
// const arrowExample = function() {  
//   return "some Value";  
// }  
const arrowExample = () => "some value";
```

- Arrow function without arguments in it. () blank brackets indicates that a function with no arguments. Starts with round bracket and argument inside the bracket.

If function is single line

```
const arrowExample = () => "some value";  
/script>
```

If multiple line

```
const tobeExceuted = (resp) => {  
  console.log(resp, "Executed", "with resp");  
  return 123;  
};
```

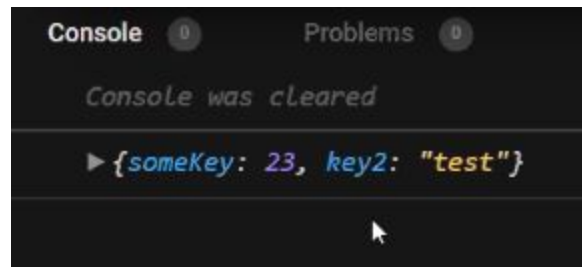
Recommended use of arrow function is for anonymous function.

Hashmap in javascript

Input code

```
const a = {};  
a["someKey"] = 23;  
const key = "key2";  
a[key] = "test";  
  
console.log(a);
```

Output



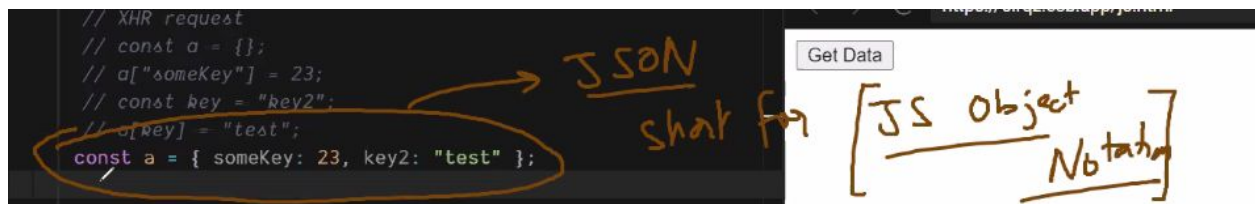
Writing in single line

```
function getData() {  
  // XHR request  
  // const a = {};  
  // a["someKey"] = 23;  
  // const key = "key2";  
  // a[key] = "test";  
  const a = { someKey: 23, key2: "test" };  
}
```

It will give the same as before, shorthand for the above two images. 'a' is a javascript object, It's a shorthand for initialization of a javascript object..

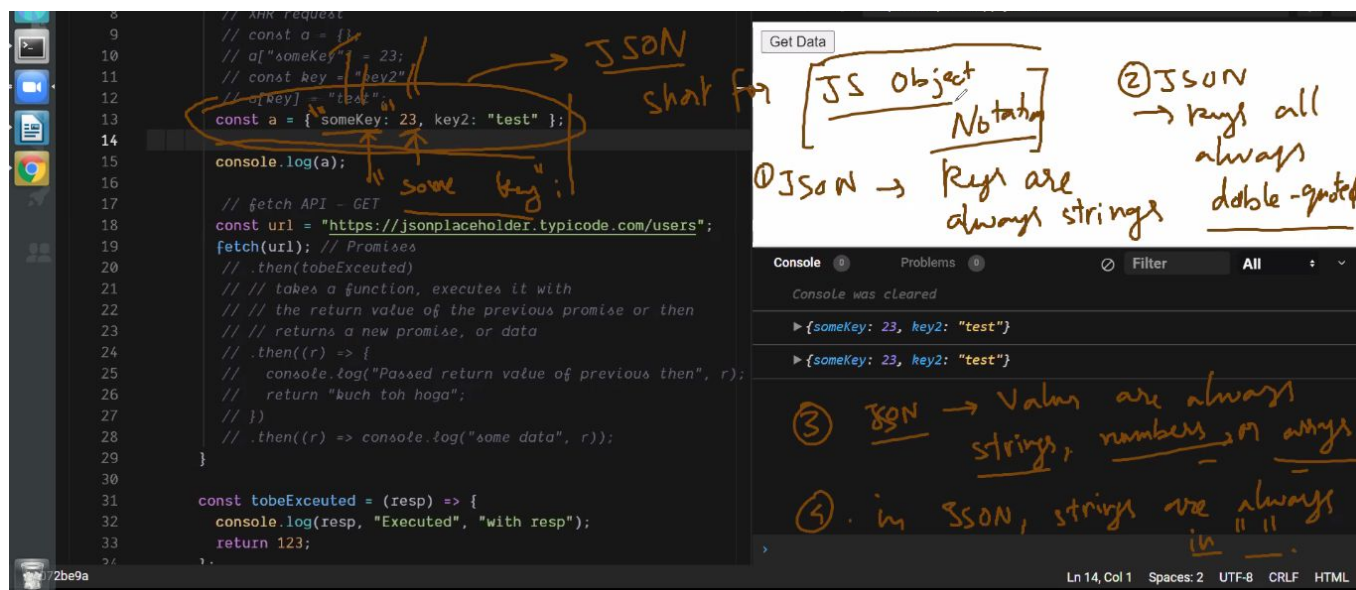
JSON

JSON-> javascript object notation



Diff between standard js object and json object

| Standard JS Object | JSON Object |
|---|---|
| Not mandatory that all keys are strings | Mandatory that all keys are strings. |
| Key can have space in it. You can use single quotes or double quotes to define a key. | No single quotes are used for keys. Only double quotes are used. |
| Key and value can be of any type | <p>Debanshu sir: Keys are always string. And values can be strings , numbers, arrays or other jsons.</p> <p>Internet: JSON can represent four primitive types (strings, numbers, booleans, and null) and two structured types (objects and arrays)</p> <p>ref:https://www.sitepoint.com/basics-json-syntax-tips/</p> |



```
{
  "id": 2,
  "name": "Ervin Howell",
  "username": "Antonette",
  "email": "Shanna@melissa.tv",
  "address": {
```

strings are always double quoted

```
    "city": "Wisokyburgh",
    "zipcode": "90566-7771",
    "geo": {
      "lat": "-43.9509",
      "lng": "-34.4618"
    }
  },
```

another json object in the json

```
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server n",
    "bs": "harness real time e-markets"
  }
},
{
  "id": 2,
  "name": "Ervin Howell",
  "username": "Antonette",
  "email": "Shanna@melissa.tv",
  "address": {
    "street": "Victor Plains",
    "suite": "Suite 879",
    "city": "Wisokyburgh",
    "zipcode": "90566-7771",
    "geo": {
      "lat": "-43.9509",
      "lng": "-34.4618"
    }
  }
}
```

The response is following the json format

- So the link responses in json format.
- Json is a js object but with some more restriction.
- 'fetch()' Will convert the json format to standard javascript object automatically.
- Response from the link is javascript array. Json object cannot start/end from/to square brackets. It always starts and ends with curly braces.

```
const a = {
  someKey: 23,
  key2: "test",
  arr: [1, 2, 3, "someArray"],
  "nested object": {
    geo: 34
  }
};
```

This a js object

How to print the second element of the array1 inside js object.

```
const a = {
  someKey: 23,
  key2: "test",
  array1: [1, 2, 3, "someArray"],
  "nested object": {
    geo: 34,
    array2: [1, 2, 3, "someArray"],
  }
};
```

Printing second element of array 1

```
console.log(a["array1"][1]);
console.log(a.array1[1]);
```

Two ways shown

Result



- 2nd syntax is shortcut, only done if key doesn't have space inside it otherwise it won't work.

How to access array 2 s third element

```
console.log(a["nested object"].array2[2]);
```

one way

```
console.log(a["nested object"]["array2"][2]);
```

second way

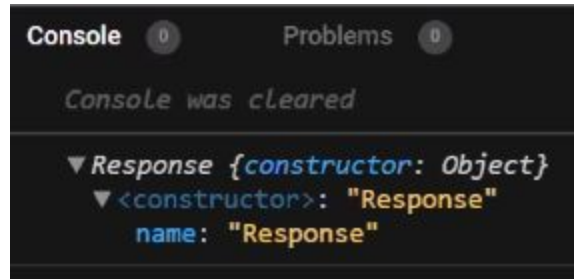
Full code for understanding access of array inside javascript

```
// a[key] = "test";  
const a = {  
  someKey: 23,  
  key2: "test",  
  array1: [1, 2, 3, "someArray"],  
  "nested object": {  
    geo: 34,  
    array2: [1, 2, 56, "someArray"]  
  }  
};  
  
console.log(a["array1"][1]);  
console.log(a.array1[1]);  
  
console.log(a["nested object"].array2[2]);  
console.log(a["nested object"]["array2"][2]);
```


Handling response from a link

Printing the response from url to console.

```
const url = "https://jsonplaceholder.typicode.com/users";
fetch(url) // Promises
  .then((r) => console.log(r));
}
```



(line 2 == line 3 to 5) `r.json()` is equivalent to commented lines 3 to 5

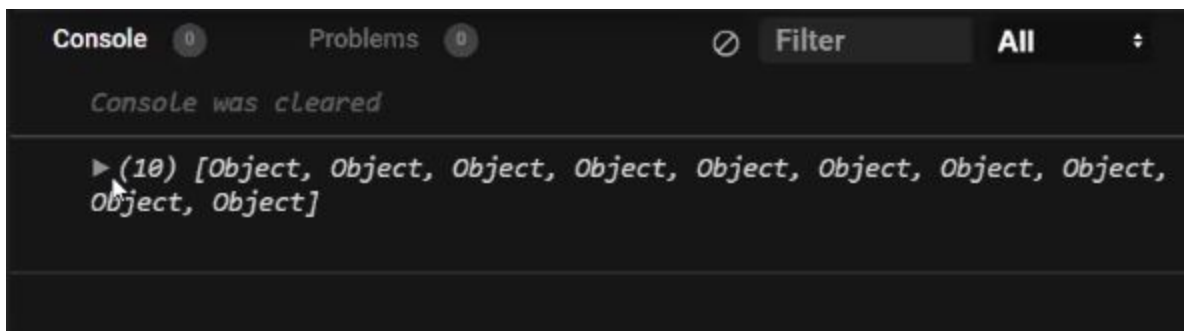
```
fetch(url) // Promises
  .then((r) => r.json());
  // .then(function(responseOfFetch) {
  //   // return responseOfFetch.json();
  // })
}
```

Javascript is automatically converting the json response you received and converts it to js object and this object is returned. Now the json object is lost because we haven't used this returned object by 'then' function.

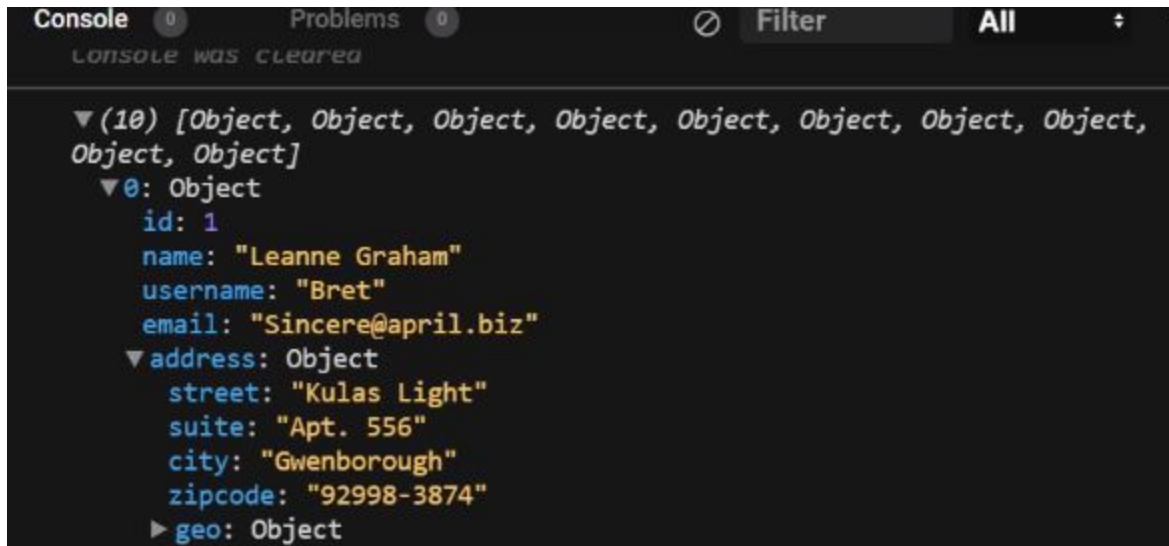
You can access response(now js object) using next then function as shown in image below. Printed the response in console.

```
// fetch API - GET
const url = "https://jsonplaceholder.typicode.com/users";
fetch(url) // Promises
  .then((r) => r.json())
  // .then(function(responseOfFetch) {
  //   // return responseOfFetch.json();
  // })
  .then((resp) => {
    console.log(resp);
  });
}
```

console



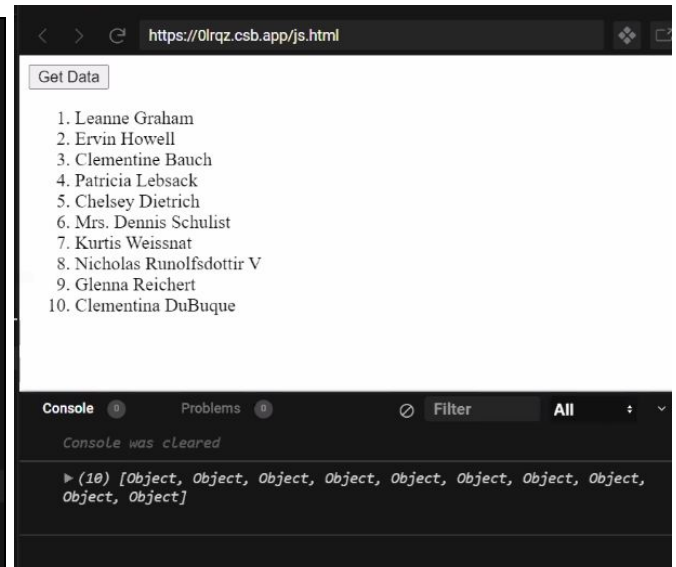
Response expanded in console



Add this data(response from url)to dom means kind of adding it to html

Creating list

```
fetch(url) // Promises
  .then((r) => r.json())
  // .then(function(responseOfFetch) {
  //   return responseOfFetch.json();
  // })
  .then((resp) => {
    console.log(resp);
    const list = document.createElement("ol");
    for (let i = 0; i < resp.length; i++) {
      const listitem = document.createElement("li");
      listitem.innerHTML = resp[i].name;
      list.appendChild(listitem);
    }
    const d1 = document.getElementById("d1");
    d1.appendChild(list);
  });
}
```



Link for debanshu sir's code.

<https://codesandbox.io/s/suspicious-snowflake-0lrqz?file=/js.html>