Async Functions

async & await Keywords

Async Keyword

Creates an Async Function

```
async function greet() {
  return "hello world!"; //returns a promise
}

let hello = async () => {}; //returns a promise
```

```
async function greet() {
  return "hello!";
}
```

```
> greet();
<· ▶ Promise {<fulfilled>: 'hello!'}
```

```
async function greet() {
  abc.abc();
  return "hello!";
}
```

```
> greet();
<· Promise {<rejected>: ReferenceError: abc is not de
  fined
  ▼     at greet (file:///Users/shradhakhapra/WebClass
  room/JS/JavaSc…} i
    ▶ [[Prototype]]: Promise
      [[PromiseState]]: "rejected"
    ▶ [[PromiseResult]]: ReferenceError: abc is not def
⊗ ▶Uncaught (in promise) ReferenceError:    app.js:30
  abc is not defined
      at greet (app.js:30:3)
      at <anonymous>:1:1
>
```

```
async function greet() {
  throw "some random error";
  return "hello!";
}
```

```
> greet();
<· ▼ Promise {<rejected>: 'some random error'}
    ▶ [[Prototype]]: Promise
      [[PromiseState]]: "rejected"
      [[PromiseResult]]: "some random error"
⊗ ▶ Uncaught (in promise) some random error   app.js:30
```

```javascript
async function greet() {
  // throw "some random error";
  return "hello!";
}

greet()
.then(()=>{
  console.log("promise was resolved");

})
.catch((err)=>{
  console.log("promise was rejected with err : ", err)
})
```

```javascript
async function greet() {
  throw "some random error";
  return "hello!";
}

greet()
  .then((result) => {
    console.log("promise was resolved");
    console.log("result was : ", result);
  })
  .catch((err) => {
    console.log("promise was rejected with err : ", err);
  });
```

Await Keyword

pauses the execution of its surrounding async function until the promise is settled (resolved or rejected)

```
async function show() {
  await colorChange("violet", 1000);
  await colorChange("indigo", 1000);
  await colorChange("green", 1000);
  await colorChange("yellow", 1000);
  await colorChange("orange", 1000);

  return "done";
}
```

```javascript
function getNum() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      let num = Math.floor(Math.random() * 10) + 1;
      console.log(num);
      resolve();
    }, 1000);
  });
}

async function demo() {
  await getNum();
  getNum();
  getNum();
}
```

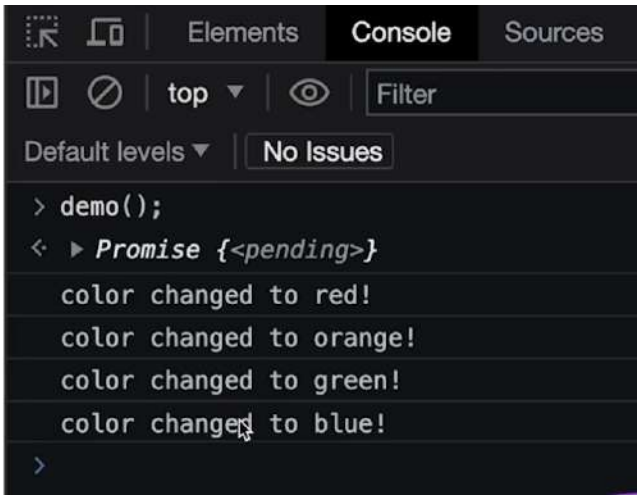
```javascript
function getNum() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      let num = Math.floor(Math.random() * 10) + 1;
      console.log(num);
      resolve();
    }, 1000);
  });
}

async function demo() {
  await getNum();
  await getNum();
  await getNum();
  await getNum();
  getNum();
```

Default levels ▼ | No Issues

```
> demo();
<· ▶ Promise {<pending>}
  7
  10
  9
  10
>
```

```javascript
h1 = document.querySelector("h1");

function changeColor(color, delay) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      h1.style.color = color;
      console.log(`color changed to ${color}!`);
      resolve("color changed!");
    }, delay);
  });
}

async function demo() {
  await changeColor("red", 1000);
  await changeColor("orange", 1000);
  await changeColor("green", 1000);
  changeColor("blue", 1000);
}
```

Apna College



Handing Rejections

```
h1 = document.querySelector("h1");

function changeColor(color, delay) {
  return new Promise((resolve, reject) =>
    setTimeout(() => {
      let num = Math.floor(Math.random() >
      if (num > 3) {
        reject("promise rejected");
      }

      h1.style.color = color;
      console.log(`color changed to ${colo
      resolve("color changed!");
    }, delay);
  });
}
```

```
> demo();
<· ▶ Promise {<pending>}
  color changed to red!                                   ap
  color changed to orange!                                ap
✕ ▶ Uncaught (in promise) promise rejected                ap
>
```

Await Keyword

Handling Rejections with Await

Await Keyword

Handling Rejections with Await

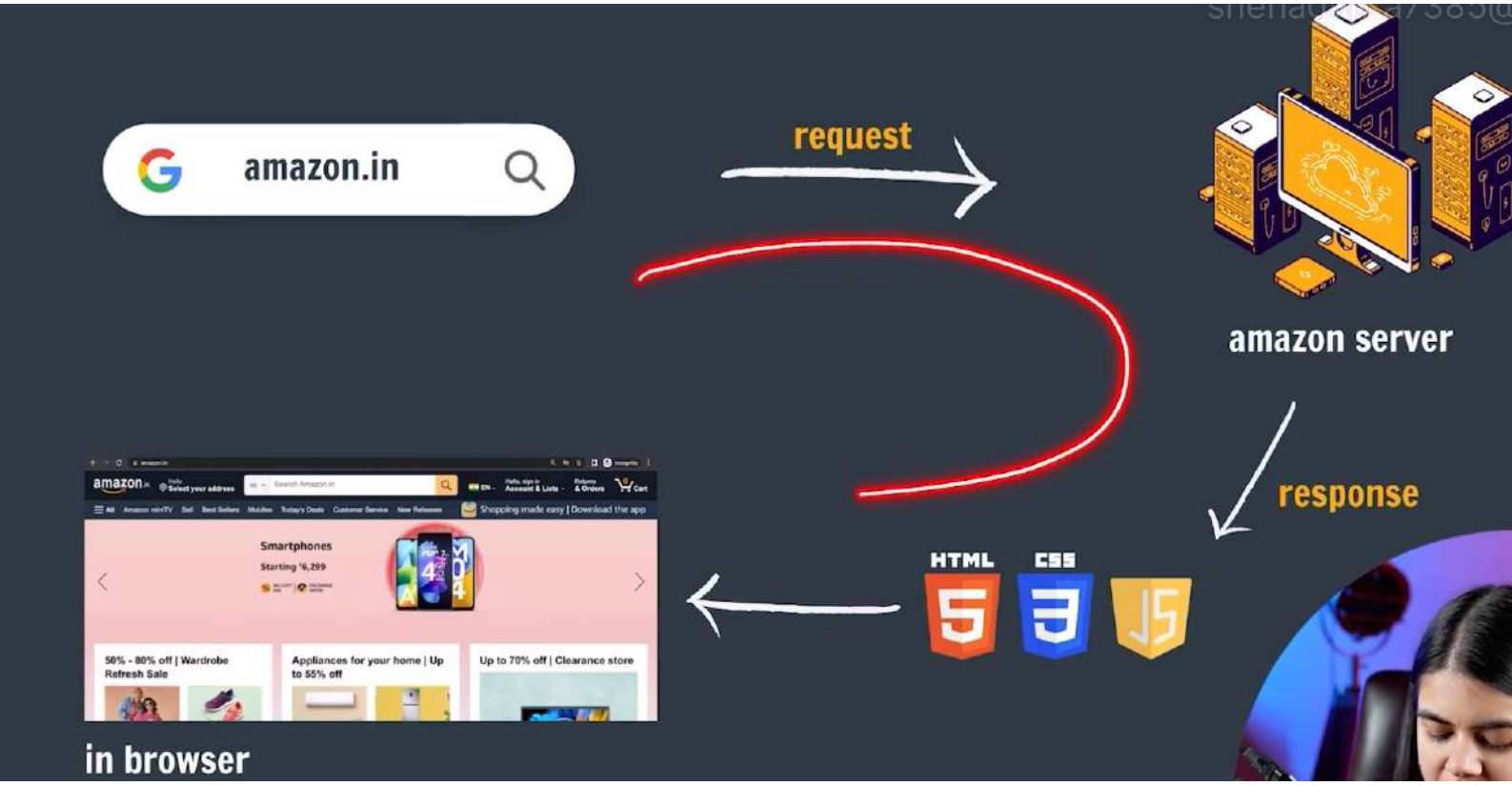
try - catch

error

```
async function demo() {
  try {
    await changeColor("red", 1000);
    await changeColor("orange", 1000);
    await changeColor("green", 1000);
    await changeColor("blue", 1000);
  } catch (err) {
    console.log("error caught");
    console.log(err);
  }
```

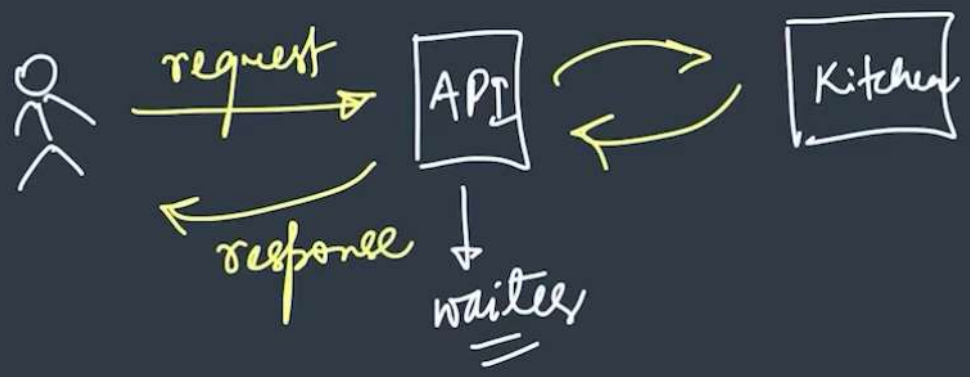
JS app.js > changeColor

```javascript
h1 = document.querySelector("h1");

function changeColor(color, delay) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      let num = Math.floor(Math.random() * 5) + 1;
      if (num > 3) {
        reject("promise rejected");
      }

      h1.style.color = color;
      console.log(`color changed to ${color}!`);
      resolve("color changed!");
    }, delay);
  });
}

async function demo() {
  try {
    await changeColor("red", 1000);
    await changeColor("orange", 1000);
    await changeColor("green", 1000);
    await changeColor("blue", 1000);
  } catch (err) {
    console.log("error caught");
    console.log(err);
  }
```

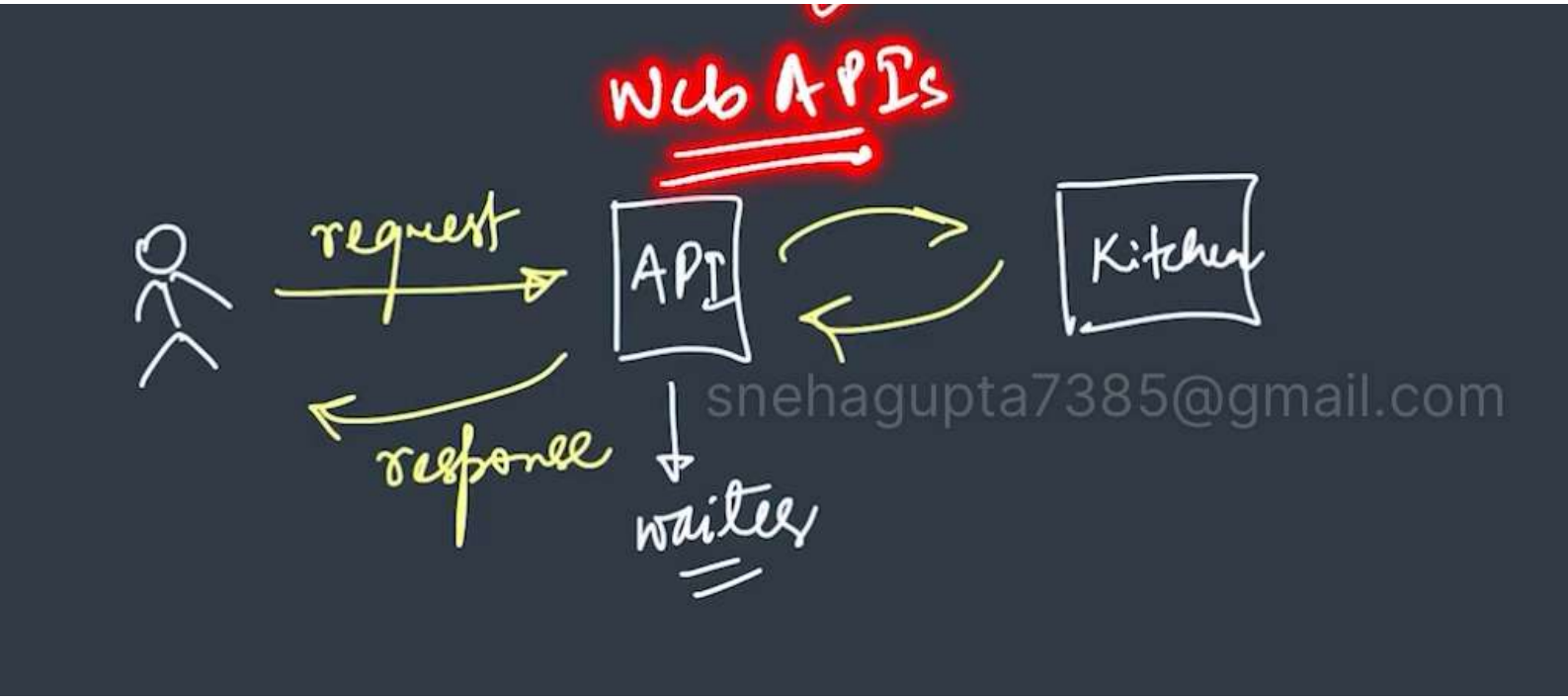

amazon.in
request
amazon server
response
HTML
CSS
JS
Smartphones
Starting ₹6,299
50% - 80% off | Wardrobe Refresh Sale
Appliances for your home | Up to 55% off
Up to 70% off | Clearance store
in browser

API

Application Programming Interface




S
W
A
P
I
S
W



API

Application Programming Interface

API

Some Random APIs

https://catfact.ninja/fact **(sends random cat facts)**

https://www.boredapi.com/api/activity **(sends an activity to do when bored)**

https://dog.ceo/api/breeds/image/random **(sends cute dog pictures)**

JSON

JavaScript Object Notation

www.json.org

XML Example

```
  <?xml version="1.0" encoding="UTF-8"?>
- <EmployeeData>
  - <employee id="34594">
        <firstName>Heather</firstName>
        <lastName>Banks</lastName>
        <hireDate>1/19/1998</hireDate>
        <deptCode>BB001</deptCode>
        <salary>72000</salary>
    </employee>
  - <employee id="34593">
        <firstName>Tina</firstName>
        <lastName>Young</lastName>
        <hireDate>4/1/2010</hireDate>
        <deptCode>BB001</deptCode>
        <salary>65000</salary>
    </employee>
  </EmployeeData>
```

JSON Example

```
1  {
2    "string": "Hi",
3    "number": 2.5,
4    "boolean": true,
5    "null": null,
6    "object": { "name": "Kyle", "age": 24 },
7    "array": ["Hello", 5, false, null, { "key": "value", "number": 6 }],
8    "arrayOfObjects": [
9      { "name": "Jerry", "age": 28 },
10     { "name": "Sally", "age": 26 }
11   ]
12 }
13
```

jsonlint.com

JSONLint - The JSON Validator

Try the New Pro

More D

```
1 {
2     "fact": "Purring does not always indicate that a cat is happy. Cats will also purr loudly when they are di
3     "length": 117
4 }
```


Validate JSON

Clear

Support JSONLint

Results

04:16 /

JSON → String

Accessing Data from JSON

- JSON.parse(data) Method

  To parse a string data into a JS object

- JSON.stringify(json) Method

  To parse a JS object data into JSON

JS app.js > ...

```
1  let jsonRes =
2    '{"fact":"Approximately 1/3 of cat owners think their pets are able to read their minds.","length":78}';
3
4  console.log(jsonRes.fact);
5
```

```
{"fact":"Approximately 1/3 of cat owners think their pets are able to  app.js:4
read their minds.","length":78}
```

app.js > ...

```js
let jsonRes =
  '{"fact":"Approximately 1/3 of cat owners think their pets are able to read their minds.","length":78}';

let validRes = JSON.parse(jsonRes);

```

```
app.js:5
{fact: 'Approximately 1/3 of cat owners think their pets are able to read the
ir minds.', length: 78}
```

JS app.js > ...

```js
let jsonRes =
  '{"fact":"Approximately 1/3 of cat owners think their pets are able to read their minds.","length":78}';

let validRes = JSON.parse(jsonRes);
console.log(validRes.fact);
```

```
Approximately 1/3 of cat owners think their pets are able to read
their minds.
```

JSON → String

Accessing Data from JSON

- **JSON.parse(data) Method**

  To parse a string data into a JS object

  json data → js object

- **JSON.stringify(json) Method**

  To parse a JS object data into JSON

  js object → json

```
let student = {
    name: "shradha",
    marks: 95,
};
```

```
> JSON.stringify(student);
<· '{"name":"shradha","marks":95}'
```

Testing API requests

Tools

- Hoppscoth
- Postman

hoppscotch.io

HOPPSCOTCH

Save My

REST

GraphQL

Realtime

Settings

My Workspace > Collections

Search

+ New

Collections are empty

Add new

GET Untitled

Select e

GET https://catfact.ninja/fact

Send

Parameters Body Headers 1 Authorization Pre-request Script Tests

Query Parameters

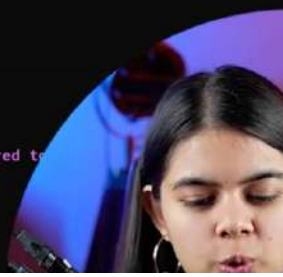
Parameter 1 Value 1

Status: 200 • OK Time: 1197 ms Size: 178 B

JSON Raw Headers 12 Test Results

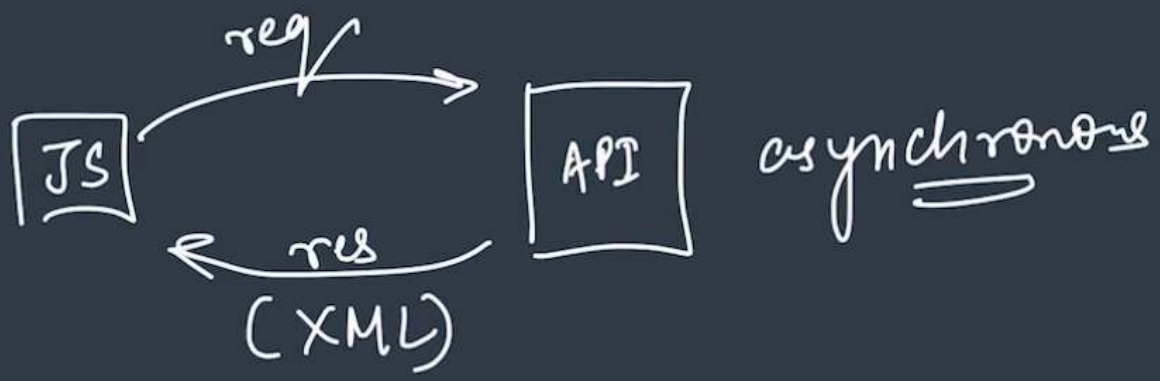
Response Body


```
1 {
2   "fact": "Cats are North America's most popular pets: there are 73 million cats compared t
    North America own a cat.",
3   "length": 149
4 }
```

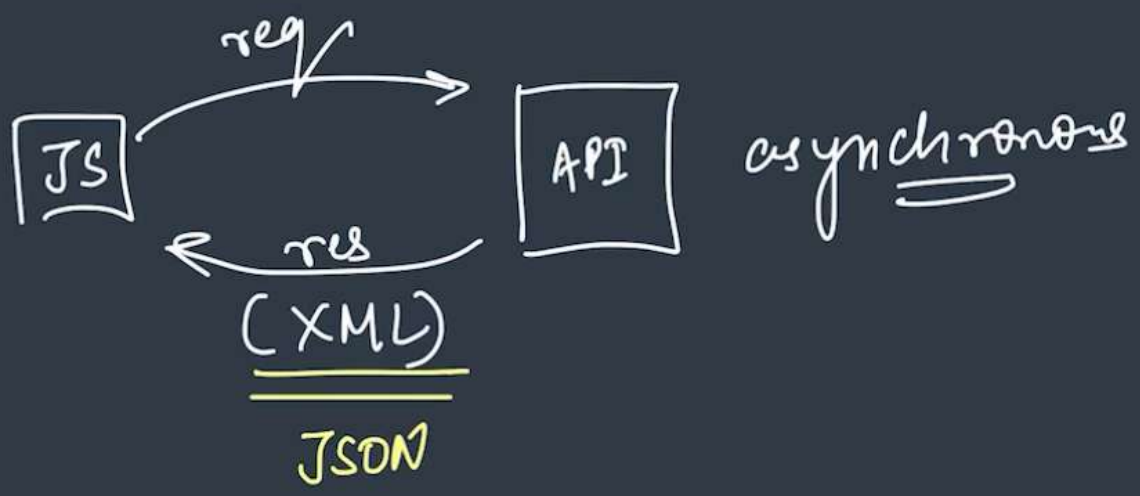
Ajax

Asynchronous JavaScript and XML



Ajax

Asynchronous JavaScript and XML



AJAJ

Http Verbs

Examples :

- GET
- POST
- DELETE

Status Codes

Examples :

- 200 - OK
- 404 - Not Found
- 400 - Bad Request
- 500 - Internal Server Error

HTTP response status codes

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

1. Informational responses (`100` – `199`)
2. Successful responses (`200` – `299`)
3. Redirection messages (`300` – `399`)
4. Client error responses (`400` – `499`)
5. Server error responses (`500` – `599`)

Add Information in URLs

Query Strings

https://www.google.com/search?q=harry+porter

Key

Value

?name=shradha&marks=95

hoppscotch.io

HOPPSCOTCH

REST

GraphQL

Realtime

Settings

My Wor... > Collections

Search

+ New

Collections are empty

Add new

GET Untitled

GET http://universities.hipolabs.com/search?name=Nigeria

Parameters Body Headers 1 Authorization Pre-request Script

Query Parameters

Parameter 1 Value 1

Status: 200 • OK Time: 658 ms Size: 909 B

JSON Raw Headers 6 Test Results

Response Body


```
38  {
39    "country": "Nigeria",
40    "domains": [
41      "unn.edu.ng"
42    ],
43    "alpha_two_code": "NG",
44    "state-province": null,
45    "web_pages": [
46      "http://www.unn.edu.ng/"
47    ],
48    "name": "University of Nigeria"
```

Http headers

header , value

Http headers

req

resp

GET https://icanhazdadjoke.com/

Parameters Body Headers 1 Authorization Pre-request Script

Header List

Accept	application/json
Header 2	Value 2

Status: 200 • OK Time: 827 ms Size: 119 B

JSON Raw Headers 22 Test Results

Response Body


```
{
  "id": "vkV0L6wcFlb",
  "joke": "Did you hear about the runner who was criticized? He
  "status": 200
}
```

GET https://icanhazdadjoke.com/

Parameters Body Headers 1 Authorization Pre-request Script Test

Header List

Accept	text/plain
Header 2	Value 2

Status: 200 • OK Time: 359 ms Size: 65 B

Raw Headers 22 Test Results

Response Body

```
1   Did you hear about the kidnapping at school? It's ok, he woke up.
```

Our First Request

using **Fetch**

{ XMLHTTPRequest obj }

fetch(url)

index.html style.css app.js

app.js > ...

```
1   let url = "https://catfact.ninja/fact";
2
3   fetch(url);
4
```

No Issues

```
> fetch(url);
<· ▼ Promise {<pending>}
    ▶ [[Prototype]]: Promise
      [[PromiseState]]: "fulfilled"
    ▶ [[PromiseResult]]: Response
>
```

app.js > catch() callback

```javascript
let url = "https://catfact.ninja/fact";

fetch(url)
  .then((response) => {
    console.log(response);
  })
  .catch((err) => {
    console.log("ERROR - ", err);
  });

```

```
app.js:5
Response {type: 'cors', url: 'https://catfact.ninja/fact', redirected: false, status: 200, ok: true, …}
  ▶ body: ReadableStream
    bodyUsed: false
  ▶ headers: Headers {}
    ok: true
    redirected: false
    status: 200
    statusText: ""
    type: "cors"
    url: "https://catfact.ninja/fact"
  ▶ [[Prototype]]: Response
```

index.html style.css app.js

JS app.js > [@] url

```
1   let url = "https://catfact.ninja/fact2";
2
3   fetch(url)
```

```
1 Issue: 1

Access to fetch at 'https://catfact.ninja/fact2' from index.html:1
origin 'null' has been blocked by CORS policy: No 'Access-Control-
Allow-Origin' header is present on the requested resource. If an
opaque response serves your needs, set the request's mode to 'no-
cors' to fetch the resource with CORS disabled.

▶GET https://catfact.ninja/fact2 net::ERR_FAILED 404  app.js:3

ERROR -  TypeError: Failed to fetch                    app.js:8
    at app.js:3:1

>
```

```
fetch(url)
  .then((res) =>
    console.log(r
    console.log(res.json()
  })
```

```
log(...data: any[]): voi
```

MDN Reference

```
▶ Response {type: 'cors', url: 'https://catfact.ninja/fact', redire
  cted: false, status: 200, ok: true, …}
                                                          app.js:6
▼ Promise {<pending>} i
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
  ▼ [[PromiseResult]]: Object
      fact: "In 1987, cats overtook dogs as the number one pet in Am
      length: 178
    ▼ [[Prototype]]: Object
```

JS app.js > then() callback

```javascript
let url = "https://catfact.ninja/fact";

fetch(url)
  .then((res) => {
    console.log(res);
    res.json().then((data) => {
      console.log(data);
    });
  })
  .catch((err) => {
    console.log("ERROR - ", err);
  });
```

```
app.js:5
▶ Response {type: 'cors', url: 'https://catfact.ninja/fact', redire
cted: false, status: 200, ok: true, …}

app.js:7
▼ {fact: 'Ginger tabby cats can have freckles around their mouths a
nd on their eyelids!', length: 77}
    fact: "Ginger tabby cats can have freckles around their mouths a
    length: 77
  ▶ [[Prototype]]: Object
```

JS app.js > ...

```javascript
let url = "https://catfact.ninja/fact";

fetch(url)
  .then((res) => {
    console.log(res);
    return res.json();
  })
  .then((data) => {
    console.log(data);
  })
  .catch((err) => {
    console.log("ERROR - ", err);
  });

```

JS app.js > then() callback

```javascript
let url = "https://catfact.ninja/fact";

fetch(url)
  .then((res) => {
    console.log(res);
    return res.json();
  })
  .then((data) => {
    console.log(data.fact);
  })
  .catch((err) => {
    console.log("ERROR - ", err);
  });
```

```
In relation to their body size, cats have the largest      app.js:
eyes of any mammal.
```

app.js > then() callback

```javascript
let url = "https://catfact.ninja/fact";

fetch(url)
  .then((res) => {
    return res.json();
  })
  .then((data) => {
    console.log("data1 = ", data.fact);
    return fetch(url);
  })
  .then((res) => {
    return res.json();
  })
  .then((data2) => {
    console.log("data2 = ", data2.fact);
  })
  .catch((err) => {
    console.log("ERROR - ", err);
  });
```

Our First Request

using Fetch with async/ await

```
async function getFacts() {
  try {
    let res1 = await fetch(url);
    let data1 = await res1.json();
    console.log("data1 - ", data1);

    let res2 = await fetch(url);
    let data2 = await res2.json();
    console.log("data2 - ", data2);
  } catch (e) {
    console.log("error : ", e);
  }
}
```

```
let url = "https://catfact.ninja/fact";

async function getFacts() {
  let res = await fetch(url);
  let data = await res.json();
  console.log(res);
}
```

```
> getFacts();
<· ▶ Promise {<pending>}

                                                    app.js:6
  {fact: 'The normal body temperature of a cat is between 10…its te
  mperature goes below 100 ° or above 103 °F.', length: 136}
    fact: "The normal body temperature of a cat is between 100.5 ° a
    length: 136
    ▶ [[Prototype]]: Object
>
```

app.js > getFacts

```javascript
let url = "https://catfact.ninja/fact2";

async function getFacts() {
  try {
    let res = await fetch(url);
    let data = await res.json();
    console.log(data.fact);
  } catch (e) {
    console.log("error - ", e);
  }
}
```

JS app.js > [⊡] url

```
1   let url = "https://catfact.ninja/fact";
2
3   async function getFacts() {
4     try {
5       let res = await fetch(url);
6       let data = await res.json();
7       console.log(data.fact);
8
9       let res2 = await fetch(url);
10      let data2 = await res2.json();
11      console.log(data2.fact);
12    } catch (e) {
13      console.log("error - ", e);
14    }
15
16    console.log("bye");
17  }
18
```

```
No Issues
> getFacts();
<· ▶ Promise {<pending>}
Cats lose almost as much fluid in the saliva while        app.js:7
grooming themselves as they do through urination.
When a family cat died in ancient Egypt, family members   app.js:11
would mourn by shaving off their eyebrows. They also held elaborate
funerals during which they drank wine and beat their breasts. The
cat was embalmed with a sculpted wooden mask and the tiny mummy was
placed in the family tomb or in a pet cemetery with tiny mummies of
mice.
bye                                                       app.js:16
>
```