



A promise represents the eventual result of an asynchronous operation

BASED ON THE PROMISE API

The fetch API is the latest standard for data retrieval.

Replaces the use of the XmlHttpRequest object







'It [scope] is an energy field created by all living things. It surrounds us.

It binds your JavaScript variables, functions and objects together'

fetch() -> Global scope.



FIRST EXAMPLE

HEAVILY USED WITH JSON

```
fetch('some url')
. then(respObject => {
    return respObject.json();
}). then(actualJSON => {
    .....
});
```

DEFAULT: A GET REQUEST

```
fetch('/api/some.json', {headers: {Accept: 'application/json'}})
   .then((res) => res.json())
   .then((json) => {
      // json contains parsed JSON
   });
```

```
fetch ('some url').then (response -> {
   if (response.status!= 200) {
      // something wrong.
   } else {
     // everything okay... continue...
```

An object that has attributes and methods to help create a fetch() request.

THE REQUEST | INTERFACE

EXAMPLE

```
const request = new
    Request("https://jsonplaceholder.typicode.com/posts/1");

fetch(request)
    .then(result => {
        result.json(); // transform result object to JSON format
}).then(json => {
        // do something with the data at this point.
});
```

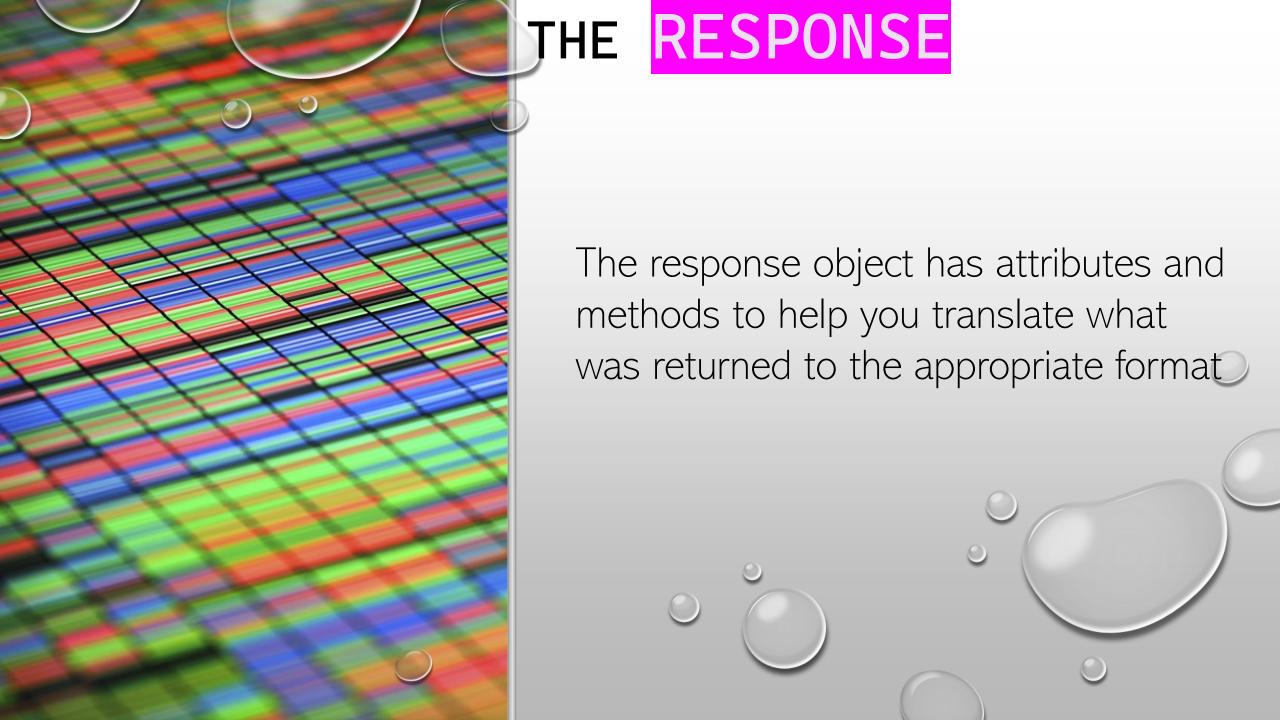
```
const token = 2x8rc-y7#4#-2*xFF;
```

```
fetch(url, headers: {
    authorization: `token ${token}`
}).then(res => {
    return res.json();
}).then(json => { ... });
```

Auth tokens

```
let content = {some: 'content'};
// The actual fetch request
fetch('some-url', {
  method: 'post',
  headers: {
    'Content-Type': 'application/json'
 },
  body: JSON.stringify(content)
// .then()...
```

```
const _headers = new Headers({
 "Content-Type": "application/json"
});
const request = new Request(url, {
    headers: _headers,
    method: "POST",
    body: {"a":1}
});
fetch (request).then (....)
```



Response: core API

RESPONSE METHODS



For Word Doc's, PDF's, movies, audio



Could be text or possibly XML



Use with JSON data

RESPONSE SIGNATION SE

basic

Same origin as request.

COrs

Different origin. You won't be able to see the data.

opaque

Different origin. Server sent no CORS headers..

You won't be able to read the response data.

EXPECTING XML?

Use the response.text() method combined with DOMParser: parseFromString().

If you do not know the data format of the response...

```
.then(response => {
 let contentType = response.headers.get('content-type')
 if (contentType.includes('application/json')) {
   return response.json()
 else if (contentType.includes('text/html')) {
   return response.text()
 else {
   // Handle other responses accordingly...
```