

FETCH()

Prof. Andrew Sheehan

*Boston University
Computer Science/MET*



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



Fetch - LS

A modern replacement for XMLHttpRequest.

Current aligned

Usage relative

Date relative

Filtered

All

Chrome

Edge ^{*}

Safari

Firefox

Opera

4-39

2-33

10-26

² 40

^{1 4} 34-38

² 27

^{2 3} 41

12-13

3.1-10

⁴ 39

^{2 3} 28

42-119

14-119

10.1-17.1

40-120

29-105

120

120

17.2

121

106

121-123

121

17.3-TP

122-124



| ‘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

```
1  fetch('http://example.com/movies.json')
2    .then(function(response) {
3      return response.json();
4    })
5    .then(function(myJson) {
6      console.log(JSON.stringify(myJson));
7    });
```

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
  });
```


RESPONSE.STATUS

```
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


POST: SENDING JSON

```
let content = {some: 'content'};

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

CREATING HEADERS WITH 2 ENTRIES

```
const headers = new Headers({  
  "Content-Type",    "application/json",  
  "Accept-Encoding", "gzip, deflate"  
});
```

A blue background with several realistic water droplets of varying sizes, some with highlights and shadows, scattered across the left side of the image.

```
const _headers = new Headers({  
  "Content-Type" : "application/json"  
});
```

```
const request = new Request(url, {  
  headers: _headers,  
  method: "POST",  
  body: {"a":1}  
});
```

```
fetch(request).then(.....)
```

The background of the slide features a vibrant, multi-colored grid pattern on the left side, transitioning into a light gray area on the right. Several realistic water droplets of various sizes are scattered across the top and bottom of the slide, adding a decorative touch.

THE RESPONSE

The response object has attributes and methods to help you translate what was returned to the appropriate format

RESPONSE METHODS

Response: core API

`.blob`

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

`.text`

Could be text or possibly XML

`.json`

Use with JSON data

RESPONSE TYPES

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...  
  }  
});
```