Chapter 55: Fetch

Options	Details
`method`	The HTTP method to use for the request. ex: `GET`, `POST`, `PUT`, `DELETE`, `HEAD`. Defaults to `GET`.
`headers`	A `Headers` object containing additional HTTP headers to include in the request.
`body`	The request payload, can be a `string` or a `FormData` object. Defaults to **`undefined`**
`cache`	The caching mode. **`default`**, `reload`, `no-cache`
`referrer`	The referrer of the request.
`mode`	`cors`, `no-cors`, `same-origin`. Defaults to `no-cors`.
`credentials`	`omit`, `same-origin`, `include`. Defaults to `omit`.
`redirect`	`follow`, `error`, `manual`. Defaults to `follow`.
`integrity`	Associated integrity metadata. Defaults to empty string.

Section 55.1: Getting JSON data

```
// get some data from stackoverflow
fetch("https://api.stackexchange.com/2.2/questions/featured?order=desc&sort=activity&site=stackover
flow")
  .then(resp => resp.json())
  .then(json => console.log(json))
  .catch(err => console.log(err));
```

Section 55.2: Set Request Headers

```
fetch('/example.json', {
    headers: new Headers({
        'Accept': 'text/plain',
        'X-Your-Custom-Header': 'example value'
    })
});
```

Section 55.3: POST Data

Posting form data

```
fetch(`/example/submit`, {
    method: 'POST',
    body: new FormData(document.getElementById('example-form'))
});
```

Posting JSON data

```
fetch(`/example/submit.json`, {
    method: 'POST',
    body: JSON.stringify({
        email: document.getElementById('example-email').value,
        comment: document.getElementById('example-comment').value
    })
});
```

Section 55.4: Send cookies

The fetch function does not send cookies by default. There are two possible ways to send cookies:

1. Only send cookies if the URL is on the same origin as the calling script.

```
fetch('/login', {
    credentials: 'same-origin'
})
```

2. Always send cookies, even for cross-origin calls.

```
fetch('https://otherdomain.com/login', {
    credentials: 'include'
})
```

Section 55.5: GlobalFetch

The GlobalFetch interface exposes the `fetch` function, which can be used to request resources.

```
fetch('/path/to/resource.json')
    .then(response => {
        if (!response.ok()) {
            throw new Error("Request failed!");
        }

        return response.json();
    })
    .then(json => {
        console.log(json);
    });
```

The resolved value is a Response Object. This Object contains the body of the response, as well as its status and headers.

Section 55.6: Using Fetch to Display Questions from the Stack Overflow API

```
const url =
      'http://api.stackexchange.com/2.2/questions?site=stackoverflow&tagged=javascript';

const questionList = document.createElement('ul');
document.body.appendChild(questionList);

const responseData = fetch(url).then(response => response.json());
responseData.then(({items, has_more, quota_max, quota_remaining}) => {
  for (const {title, score, owner, link, answer_count} of items) {
    const listItem = document.createElement('li');
    questionList.appendChild(listItem);
    const a = document.createElement('a');
    listItem.appendChild(a);
    a.href = link;
    a.textContent = `[${score}] ${title} (by ${owner.display_name || 'somebody'})`
  }
});
```