Cheerio tutorial

last modified June 24, 2022

Cheerio tutorial shows how to do web scraping in JavaScript with Cheerio module. Cheerio implements the core of jQuery designed for the server.

Cheerio

Cheerio is a fast, flexible, and lean implementation of core jQuery designed specifically for the server.

It parses markup and provides an API for traversing/manipulating the resulting data structure. I

Cheerio selectors

In Cherrion, we use selectors to select tags of an HTML document. The selector syntax was borrowed from jQuery.

The following is a partial list of available selectors:

- $("*") — selects all elements

- $("h2, div, p") — selects all `<h2>, <div>, <p>` elements
- $("li:first") — selects the first `<li>` element
- $("li:last") — selects the last `<li>` element
- $("li:even") — selects all even `<li>` elements
- $("li:odd") — selects all odd `<li>` elements
- $(":empty") — selects all elements that are empty
- $(":focus") — selects the element that currently has focus

Installing Cheerio and Node fetch

We install cheerio and node-fetch modules. Node Fetch is a package to create requests in JavaScript.

```
$ nodejs -v
v18.2.0
```

We use Node version 18.2.0.

```
$ npm i cheerio
$ npm i node-fetch
```

We install `cheerio` and `node-fetch` packages.

Cheerio title

In the first example, we get the title of the document.

main.js

```
import fetch from 'node-fetch';
import { load } from 'cheerio';

const url = 'http://webcode.me';

const response = await fetch(url);
const body = await response.text();

let $ = load(body);

let title = $('title');
console.log(title.text());
```

The example retrieves the title of an HTML document.

We include `cheerio` and `node-fetch` modules.

```
const url = 'http://webcode.me';

const response = await fetch(url);
const body = await response.text();
```

We create a GET request to the web page. We get the response object and from the response object, we get its body.

```
let $ = cheerio.load(body);
```

First, we load the HTML document. To mimic jQuery, we use the `$` variable.

```
let title = $('title');
```

The selector returns the `title` tag.

```
console.log(title.text());
```

With the `text` method, we get the text of the `title` tag.

```
$ node main.js
My html page
```

The example prints the title of the document.

Cheerio get parent element

The parent element is retrieved with `parent`.

main.js

```
import fetch from 'node-fetch';
import { load } from 'cheerio';

const url = 'http://webcode.me';

const response = await fetch(url);
const body = await response.text();

let $ = load(body);

let title = $('title');
let parentEl = title.parent();
```

```
$ node main.js
head
```

Cheerio first & last element

The first element of a cheerio object can be found with `first`, the last element with `last`.

main.js

```
import fetch from 'node-fetch';
import { load } from 'cheerio';

const url = 'http://webcode.me';

const response = await fetch(url);
const body = await response.text();

let $ = load(body);

let head = $('head');

let fel = head.children().first();
let lel = head.children().last();

console.log(fel.get(0).tagName);
console.log(lel.get(0).tagName);
```

The example prints the first and last element of the `main` tag.

```
let head = $('head');
```

We select the `head` tag.

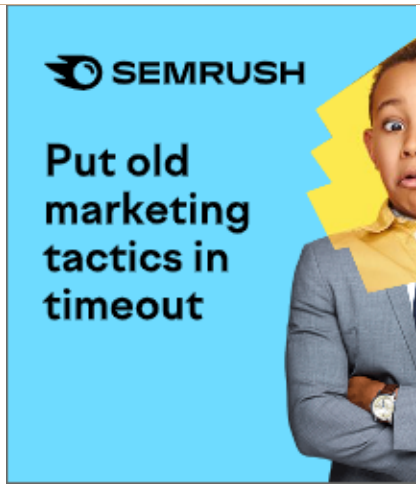
```
let fel = head.children().first();
let lel = head.children().last();
```

We get the first and the last element from the `main` children.

```
console.log(fel.get(0).tagName);
console.log(lel.get(0).tagName);
```

We find out the tag names.

```
$ node main.js
meta
title
```



Cheerio add element

The append method adds a new element at the end of the specified tag.

main.js

```
import fetch from 'node-fetch';
import { load } from 'cheerio';

const url = 'http://webcode.me';

const response = await fetch(url);
const body = await response.text();

let $ = load(body);

let bodyEl = $('body');

bodyEl.append('<p>An old falcon</p>');

let content = $('body').html();
let items = content.split('\n');

items.forEach((e) => {
    console.log(e.trim());
});
```

In the example, we add a paragraph to the body tag and print it to the console.

```
bodyEl.append('<p>An old falcon</p>');
```

We append a new paragraph.

```
let content = $('body').html();
```

```
let items = content.split('\n');

items.forEach((e) => {
    console.log(e.trim());
});
```

We print the contents and strip some whitespace.

Cheerio insert after element

With `after`, we can insert an element after a tag.

main.js

```
import fetch from 'node-fetch';
import { load } from 'cheerio';

const url = 'http://example.com'

const response = await fetch(url);
const body = await response.text();

let $ = load(body);

$('div').after('<footer>This is a footer</footer>')

console.log($.html());
```

In the example, we insert a `footer` element after the `div` element.

Cheerio loop over elements

With `each`, we can loop over elements.

main.js

```
import fetch from 'node-fetch';
import { load } from 'cheerio';

const url = 'http://webcode.me/countries.html'

const response = await fetch(url);
const body = await response.text();

let $ = load(body);

$('tr').each((_, e) => {
```

The example loops over `tr` tags and prints the text of the elements.

```
$ node main.js
Id Name Population
1 China 1382050000
2 India 1313210000
3 USA 324666000
4 Indonesia 260581000
5 Brazil 207221000
6 Pakistan 196626000
...
```

Cherio get nth element

With the `nth-child` pseudo-class, we can select the nth element of a certain tag.

main.js

```
import fetch from 'node-fetch';
import { load } from 'cheerio';

const url = 'http://webcode.me/countries.html'

const response = await fetch(url);
const body = await response.text();

let $ = load(body);

let el9 = $('tr:nth-child(9)');
let res = el9.text().trim();

console.log(res.replace(/(\s+)/g, ' '));
```

In the example, we select the ninth element of the list of most populated countries in the world.

```
$ node main.js
9 Russia 146838000
```

Cheerio get element attributes

Attributes can be retrieved with `attr` function.

main.js

```
import fetch from 'node-fetch';
import { load } from 'cheerio';

const url = 'http://webcode.me';
```

```
let $ = load(body);

let lnEl = $('link');
let attrs = lnEl.attr();

console.log(attrs.rel);
console.log(attrs.href);
```

In the example, we get the `rel` and `href` attributes of the `link` tag.

```
$ node main.js
stylesheet
format.css
```

Cheerio filter elements

We can use `filter` to apply a filter on the elements.

main.js

```
import fetch from 'node-fetch';
import { load } from 'cheerio';

const url = 'http://webcode.me';

const response = await fetch(url);
const body = await response.text();

let $ = load(body);
let all = $('*');

let res = all.filter((i, e) => {

    return $(e).children().length === 2;
});

let items = res.get();

items.forEach(e => {
    console.log(e.name);
});
```

In the example, we find out all elements of the document that contain two children.

```
let all = $('*');
```

The * selector selects all elements.

```
let res = all.filter((i, e) => {
```

On the retrieved elements, we apply a filter. An element is included in the filtered list only if it contains exactly two children.

```
let items = res.get();

items.forEach(e => {
    console.log(e.name);
});
```

We go through the filtered list and print the names of the elements.

```
$ node main.js
html
body
```

In this article, we have done web scraping in JavaScript with the Cheerio library.

List all JavaScript tutorials.

© 2007 - 2022 Jan Bodnar admin(at)zetcode.com