Module 0342: Using cheerio and regex to change all hrefs to full URLs

Tak Auyeung, Ph.D.

November 30, 2021

Contents

1 About this module

- Prerequisites:
- Objectives: This module combines the use of cheerio and regular expressions to find all relative URLs and update every one to a fully specified URL

2 Objectives

In this module, we are learning how to use `cheerio` to utilize regular expressions to find all `href` attributes that specify a relative path. For each relative path found, the relative path should be changed to a fully qualified absolute path.

To save you the trouble of starting from scratch, you can download appBS.js and use it as a starting point. Of course, if you have a working script from the "cheerio to the rescue" lab, you can use your own code as a starting point. Be sure to copy the old file to a new name so you do not lose the old file or to change the timestamp of the old file.

You can also download getThisCheerio.js, make a copy, and modify this script for testing purposes. Generally speaking, it is more convenient to test features in a script that does not respond to HTTP requests.

3 How to find all the 'href' attributes

`cheerio`, like JQuery, has a 'selector' that can be used to get all the elements based on the presence of an attribute.

In the previous porject, we utilized a selector to find an element based on the type of the element ('link'). This time, however, we are locating the elements based on the presence of an 'href' attribute.

In the previous project, to find all the 'link' elements, the select was specified as `'link'`. In this project, to find all elements that have a `href` attribute, the selector is `'[href]'`.

Modify getThisCheerio.js to locate elements with an href attribute, and print the number of such elements. The application of a selector returns an array of elements, and every array has a `length` member.

4 Iterating elements in an array

Unlike a more traditional programming language like C/C++, Javascript provides a very handy way to go through elements in an array.

There is the `forEach` prototype method that is defined for every array. However, because `forEach` is a method of an array, it is technically not a control structure. A control structure is a programming language feature that specifies the flow of execution of a program. Because of this, `forEach` cannot handle `async` expressions.

An alternative is the `for(... of ...)` control structure. For example, if `elements` is the name of an array, then the following code can be used to iterate each item in the array:

```
001 for (let item of elements)
002 {
003   // do something with item here
004 }
```

Because we know each `item` of the array returned by applying a selector is an element with a `href` attribute, you can test your code by specifying `console.log(item.attribs['href'])` as the "do something with item here".

5 Find out which `href` to modify

In the previous step, you find that not all the `href` attributes need to be turned into full URL paths. Some are already full URLs. Even more interesting, some specify a bookmark (beginning with a pound # symbol).

This is where we need to utilize the regular expression from the last step of the "just your regular regular expression" lab. Because the `href` attribute of an element is a string, the `match` method can be applied to check to see if a `href` is a full URL or not.

Use a conditional statement to filter and only show the `href` attributes that are *not* full URLs.

6 Turning a relative `href` into an absolute one

If we already know the full URL of the web page that contains the relative URL paths, then we can simply insert the path of the web page before every relative `href`. In this case, we can insert `https://power.arc.losrios.edu/` in front of all the relative `href`.

6.1 Finding the path of the container page

However, this simplistic method only works when we make an assumption about the URL of the containing page. The containing page can be specified in many different ways. For example, the following are all equivalent:

- `https://power.arc.losrios.edu`
- `https://power.arc.losrios.edu/`
- `https://power.arc.losrios.edu/index.pl`

What can be helpful is to apply the regular expression to each of these strings, and then try to figure out the logic to find the path (without the file `index.pl`).

`"https://power.arc.losrios.edu".match(/^https?:\/\/(\w+(\.\w+)*)(:\d+)?((\/\~?(\w|\.)+)(\/(\w|\.)+)*\/)?/)` yields the following outcome (comments added after the fact):

```
[
  'https://power.arc.losrios.edu',
  'power.arc.losrios.edu',
  '.edu',
  undefined, // port number
  undefined, // path
  undefined,
  undefined,
  undefined,
  undefined,
  index: 0,
  input: 'https://power.arc.losrios.edu',
  groups: undefined
]
```

Interestingly, the other two URLs give us exactly the same result. Note how element 4 (counting from zero) is `undefined`, indicating we are referring to the root of a server.