

EECS 498 W26 System Design of a Search Engine

04 HTML parser

Due Sunday, February 1, 2026, 11:59 pm

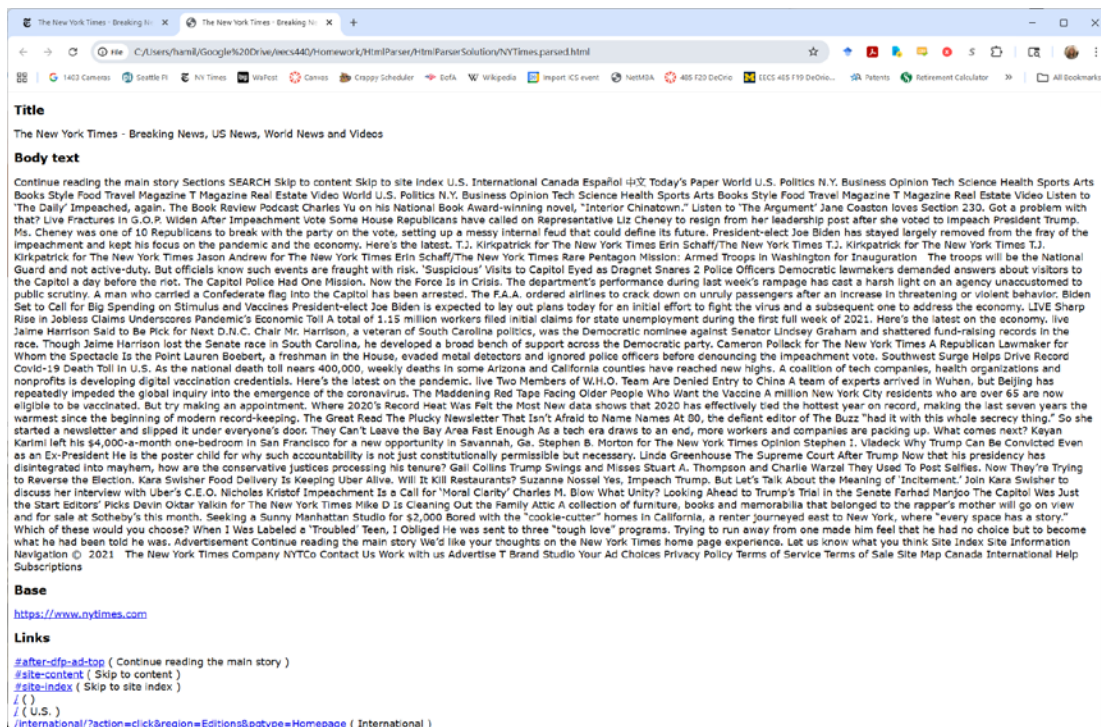
The purpose of this assignment is to get you started parsing the content on an HTML page, extracting the title, body text, any base, and the links and anchor text. It makes no attempt to run scripts or build an actual DOM (Document Object Model) of the page.

Starter files

You are given these starter files in `HtmlParserStarter.zip`.

<code>HtmlParser.h</code> <code>HtmlParser.cpp</code>	Starter files for an <code>HtmlParser</code> class that can extract the title, body text, base, and the links and anchor text on a possibly broken HTML page.
<code>HtmlTags.h</code>	A sorted array of HTML tags and desired actions. (This file should not be modified.)
<code>HtmlTags.cpp</code>	A starter file for a <code>LookupPossibleTag</code> function to search the array of tags.
<code>ParsePage.cpp</code>	A simple but complete demo program with various command line options to exercise your HTML parser for testing.
<code>NYTimes.html</code>	A snapshot of the <code>NYTimes.com</code> homepage taken January 14, 2021.
<code>NYTimes.parsed.html</code>	The expected output from <code>./ParseHtml NYTimes.html</code> with a fully working HTML parser.
<code>Broken.html</code>	A short file with some broken HTML.
<code>Broken.parsed.html</code>	The expected output from <code>./ParseHtml broken.html</code> with a fully working HTML parser.
<code>RandomPages.html</code>	A long HTML file with a concatenation of random web pages in various languages from around the world.
<code>RandomPages.parsed.html</code>	The expected output from <code>./ParseHtml RandomPages.html</code> with a fully working HTML parser.
<code>Makefile</code>	A makefile to build and test your HTML parser with these test pages.

Here is the NYTimes page and how it looks when correctly parsed.



The assignment

You are to implement the `LookupPossibleTag( )` function defined in `HtmlTags.h` and the `HtmlParser` class defined in `HtmlParser.h`. You may add code to `HtmlTags.cpp`, `HtmlParser.h` and `HtmlParser.cpp`.

Upload your `HtmlTags.cpp`, `HtmlParser.h` and `HtmlParser.cpp` files to the autograder at <https://autograder.io/web/project/3493>.

Hints

Watch for buffer overruns and memory leaks.

To hit the performance target:

1. Avoid doing things like copying or lower-casing characters until you know you have to, e.g., when you know you've found an actual word or possible tag.
2. Use character pointers rather than indexes into character arrays.
3. Don't collect characters by adding them one-by-one to a string. Create a string only when you've found the begin and end of word with pointers and have decided to add it to one of the lists of words.
4. Avoid writing loops that require unnecessary calculations in the loop. For example, instead of

```
for ( char *p = buffer; p < buffer + length; p++ ) ...
```


write

```
for ( char *p = buffer, *end = buffer + length; p < end; p++ ) ...
```
5. Don't new stuff on the heap that could be on the stack.
6. The tags array is sorted so you can binary search.