HTML Parser

Submitted by,

ABHILASH J PILLAI

ARCHANA E A

LAKSHMI SURESH

NITHIL GIGI

SUJITH C MOHAN

ABSTRACT

There are various platform□s which generate content that we want to export from HTML to PDF. For example, Marketing Centerproduct lets users design print-ready marketing materials such as flyers, brochures, postcards and notecards. They are designed in HTML / JavaScript / CSS, and need to be converted to a PDF that we can send.

Previously, SaaS service called DocRaptor was used for this purpose. We decided to build our own internal service for several reasons:

- **Rendering Fidelity:** DocRaptor uses the PrinceXML library, which renders HTML somewhat differently than a browser and doesn□t support the latest HTML + CSS standards. This was by far the main reason □ we wanted to render the PDF as close as possible to what the user was seeing.
- **Performance:** by keeping requests inside our own cloud infrastructure, we could avoid sending several megabytes of data across the open internet.
- **Cost:** we could avoid paying DocRaptor□s monthly fee.
- **Security:** keeping requests inside our own cloud also has the advantage of avoiding sending our data to third parties, as well as exposing our data in flight.

INTRODUCTION

A computer program that parses content is called a parser. HTML parsing is basically: taking in HTML code and extracting relevant information like the title of the page, paragraphs in the page, headings in the page, links, bold text etc.

Parsing a document means translating it to a structure the code can use. The result of parsing is usually a tree of nodes that represent the structure of the document. This is called a parse tree or a syntax tree.

Parsing is based on the syntax rules the document obeys: the language or format it was written in. Every format you can parse must have deterministic grammar consisting of vocabulary and syntax rules. It is called a context free grammar.

IMPLEMENTATION

A rendering engine will start getting the contents of the requested document.After that, this is the basic flow of the rendering engine:

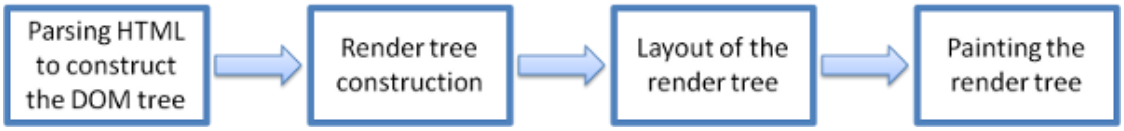


Figure : Rendering engine basic flow

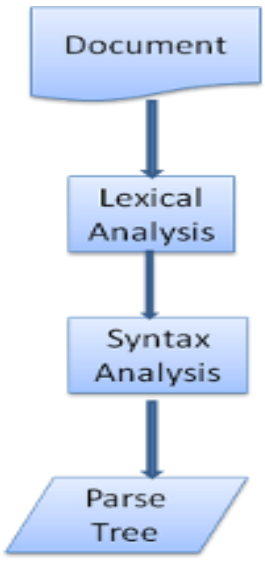
The rendering engine will start parsing the HTML document and convert elements to DOM nodes in a tree.

Parsing can be separated into two sub processes: lexical analysis and syntax analysis.

Lexical analysis is the process of breaking the input into tokens. Tokens are the language vocabulary: the collection of valid building blocks.

Syntax analysis is the applying of the language syntax rules.

Parsers usually divide the work between two components: the lexer (sometimes called tokenizer) that is responsible for breaking the input into valid tokens, and the parser that is responsible for constructing the parse tree by analyzing the document structure according to the language syntax rules. The lexer knows how to strip irrelevant characters like white spaces and line breaks.



The parsing process is iterative. The parser will usually ask the lexer for a new token and try to match the token with one of the syntax rules. If a rule is matched, a node corresponding to the token will be added to the parse tree and the parser will ask for another token.

If no rule matches, the parser will store the token internally, and keep asking for tokens until a rule matching all the internally stored tokens is found. If no rule is found then the parser will raise an exception. This means the document was not valid and contained syntax errors.

CODE :

```
from html.parser import HTMLParser
from xml.etree import ElementTree
class NaiveHTMLParser(HTMLParser):
    def __init__(self):
        self.root = None
        self.tree = []
        HTMLParser.__init__(self)

    def feed(self, data):
        HTMLParser.feed(self, data)
        return self.root

    def handle_starttag(self, tag, attrs):
        print("Encountered a start tag:", tag)
        if len(self.tree) == 0:
            element = ElementTree.Element(tag, dict(self.__filter_attrs(attrs)))
            self.tree.append(element)
            self.root = element
        else:
            element = ElementTree.SubElement(self.tree[-1], tag,
dict(self.__filter_attrs(attrs)))
            self.tree.append(element)
```

```
    def handle_endtag(self, tag):
        print("Encountered an end tag:", tag)
        self.tree.pop()

    def handle_startendtag(self, tag, attrs):
        self.handle_starttag(tag, attrs)
        self.handle_endtag(tag)
        pass

    def handle_data(self, data):
        fo.write(data)
        print("Encountered some data:", data)
        if self.tree:
            self.tree[-1].text = data

    def get_root_element(self):
        return self.root

    def __filter_attrs(self, attrs):
        return filter(lambda x: x[0] and x[1], attrs) if attrs else []

f = open("test1.html","r")
html = f.read()
print("\nINPUT")
print("------------------------")
```

```
print(html)
print("\n\nOUTPUT")
print("------------------------")

if __name__ == "__main__":
    fo=open("new.txt","w")
    parser = NaiveHTMLParser()
    root = parser.feed(html)
    parser.close()

    print("\n")
    print(root.find('head/title').text)
    for a in root.findall('.//a'):
        print(a.text)
        print(a.get('href'))
    for a in root.findall('.//div'):
        print(a.text)
    for a in root.findall('.//p'):
        print(a.text)
    for a in root.findall('.//b'):
        print(a.text)
```

SAMPLE INPUT

```
----------------------
<HTML>
<HEAD>
  <TITLE>Your Title Here</TITLE>
  <style type=text/css>
      div
      {
        color: white;
        background-color: 009900;
        margin: 2px;
        font-size: 25px;
      }
      </style>
  </HEAD>
  <BODY BGCOLOR="FFFFFF">
  <CENTER><IMG SRC="clouds.jpg" ALIGN="BOTTOM"> </CENTER>
  <HR>
  <a href="https://github.com/Sujithcmohan63/HTMLparser">Link Name</a>
  is a link to github repository
  <div > div tag   </div>
  <H1>This is a Header</H1>
  <H2>This is a Medium Header</H2>
```

<P> This is a new paragraph!

<P> <B>This is a new paragraph!</B>

 <B><I>This is a new sentence without a paragraph break, in bold italics.</I></B>

<HR>

</BODY>

</HTML>

OUTPUT :

Encountered a start tag: html

Encountered some data:

Encountered a start tag: head

Encountered some data:

Encountered a start tag: title

Encountered some data: Your Title Here

Encountered an end tag: title

Encountered some data:

Encountered a start tag: style

Encountered some data:

```
    div
    {
      color: white;
      background-color: 009900;
      margin: 2px;
      font-size: 25px;
    }
```

Encountered an end tag: style

Encountered some data:

Encountered an end tag: head

Encountered some data:

Encountered a start tag: body

Encountered some data:

Encountered a start tag: center

Encountered a start tag: img

Encountered some data:

Encountered an end tag: center

Encountered some data:

Encountered a start tag: hr

Encountered some data:

Encountered a start tag: a

Encountered some data: Link Name

Encountered an end tag: a

Encountered some data:

is a link to github repository

Encountered a start tag: div

Encountered some data: div tag

Encountered an end tag: div

Encountered some data:

Encountered a start tag: h1

Encountered some data: This is a Header

Encountered an end tag: h1

Encountered some data:

Encountered a start tag: h2

Encountered some data: This is a Medium Header

Encountered an end tag: h2

Encountered some data:

Encountered a start tag: p

Encountered some data: This is a new paragraph!

Encountered a start tag: p

Encountered some data:

Encountered a start tag: b

Encountered some data: This is a new paragraph!

Encountered an end tag: b

Encountered some data:

Encountered a start tag: br

Encountered some data:

Encountered a start tag: b

Encountered a start tag: i

Encountered some data: This is a new sentence without a paragraph break, in bold italics.

Encountered an end tag: i

Encountered an end tag: b

Encountered some data:

Encountered a start tag: hr

Encountered some data:

Encountered an end tag: body

Encountered some data:

Encountered an end tag: html

Your Title Here

Link Name

https://github.com/Sujithcmohan63/HTMLparser

div tag

This is a new paragraph!

This is a new paragraph!