HTML TAGS: <img> and <a>

COMPILER DESIGN

Submitted To :-

Dr. Ankit Rajpal

Submitted by :-

Divya Solanki (18)

Vandana Yadav (61)

2022

MCA Sem IV

Department of Computer Science

University of Delhi

INDEX

Problem Statement

To create a parser using lex and yacc to parse the HTML tags <img> and <a>

Syntax of HTML Tags

<img> Tag

Syntax:

<img src="image_name.jpg" alt="text" >

The <img> tag is used to embed an image in an HTML page.

The <img> tag has two required attributes:

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image, if the image for some reason cannot be displayed

Attributes:

- src
- alt
- crossorigin
- height
- width
- ismap
- loading
- longdesc
- referrerpolicy
- sizes
- srcset
- usemap

<a> Tag

Syntax:

```
<a href="https://www.xyzz.com">Visit xyzz.com</a>
```

The <a> tag defines a hyperlink, which is used to link from one page to another.

If the <a> tag has no href attribute, it is only a placeholder for a hyperlink.

Attributes:

- **charset**
- **download**
- **hreflang**
- **media**
- **name**
- **rel**
- **shape**
- **type**
- **target**
- **rev**
- **ping**

Assumptions

1. Assuming attributes are in lower case only for simplicity.
2. Compulsory attribute should come before non-compulsory attribute.
3. The case of nested tag is covered here.
4. Attribute values must be in either double quotes or single quotes.
5. Only <a> tag can have nested <a> or <img> tags.

Test Cases for <img> tag (Valid)

1. A img tag used with all compulsory attributes and other attributes.

Eg:

<img src=”abc.png” alt=”abc” height=”100” width=”100”>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

Enter the string : <img src="abc.png" alt="abc" height="100" width="100">

[Parsing]: <

[Parsing]: img

[Parsing] src="abc.png"

[Parsing]: alt="abc"

[Parsing]: height="100"

[Parsing]: width="100"

[Parsing]: >

Valid syntax ...
```

2. A img tag with only compulsory attribute.

Eg:

<img src=”abc.png” alt=”abc”></img>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

Enter the string : <img src="abc.png" alt="abc"></img>

[Parsing]: <

[Parsing]: img

[Parsing] src="abc.png"

[Parsing]: alt="abc"

[Parsing]: Image end tag: ></img>

Valid syntax ...
```

3. A img tag can end in three ways
   a. End with "></img>" [No whitespaes in between]
      Eg. <img src="abc.png" alt="abc" height="100" ismap="1"> </img>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

Enter the string : <img src="abc.png" alt="abc" height="100" ismap="1"> </img>

[Parsing]: <

[Parsing]: img

[Parsing] src="abc.png"

[Parsing]: alt="abc"

[Parsing]: height="100"

[Parsing]: ismap="1"

[Parsing]: Image end tag: > </img>

Valid syntax ...
```

   b. End with "/>"
      Eg.

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

Enter the string : <img src="abc.png" alt="abc" height="100" ismap="1"/>

[Parsing]: <

[Parsing]: img

[Parsing] src="abc.png"

[Parsing]: alt="abc"

[Parsing]: height="100"

[Parsing]: ismap="1"

[Parsing]: />

Valid syntax ...
```

c. End with “>”
Eg. <img src=”abc.png” alt=”abc” height=”100” ismap=”1”>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

Enter the string : <img src="abc.png" alt="abc" height="100" ismap="1">

[Parsing]: <

[Parsing]: img

[Parsing] src="abc.png"

[Parsing]: alt="abc"

[Parsing]: height="100"

[Parsing]: ismap="1"

[Parsing]: >

Valid syntax ...
```

Test Cases for <img> tag (Invalid)

1. A img tag with no attribute.
   Eg:
   <img>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <img>

 [Parsing]: <

 [Parsing]: img

 [Parsing]: >

 Invalid syntax : Error : syntax error
```

2. A img tag with compulsory attribute missing
   Eg:
   <img height=”100” weight=”100”>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <img height="100" weight="100">

 [Parsing]: <

 [Parsing]: img

 [Parsing]: height="100"

 Invalid syntax : Error : syntax error
```

3. A img tag with one compulsory attribute missing
   Eg:
   <img src=”abc”>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <img src="abc">

 [Parsing]: <

 [Parsing]: img

 [Parsing] src="abc"

 [Parsing]: >

 Invalid syntax : Error : syntax error
```

4. A img tag with all compulsory or optional attributes but invalid ending
   Eg
   a. End tag is not matched with starting tag
      <img src=”abc.png” alt=”abc” height=”100” width=”100”>


```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <img src="abc.png" alt="abc" height="100" width="100"></b>

 [Parsing]: <

 [Parsing]: img

 [Parsing] src="abc.png"

 [Parsing]: alt="abc"

 [Parsing]: height="100"

 [Parsing]: width="100"

 [Parsing]: >

 [Parsing]: </b>

 Invalid syntax : Error : Unknown End Tag.
```

5. A img tag with missing attribute values
   Eg:
   <img src= alt=>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <img src= alt=>

 [Parsing]: <

 [Parsing]: img

 [Parsing]: src

 Invalid syntax : Error : Unknown text.
```

6. A img tag with content inside tags
   Eg:
   <img src=”abc.png” alt=”abc”>Hello</img>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <img src="abc.png"  alt="abc">Hello</img>

 [Parsing]: <

 [Parsing]: img

 [Parsing] src="abc.png"

 [Parsing]: alt="abc"

 [Parsing]: >

 [Parsing]: Hello

 Invalid syntax : Error : Unknown text.
```

Test Cases for <a> tag (Valid)

1. A <a>tag used with attributes and same start and end tag.
   Eg:


```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <a href="www.google.com"></a>

 [Parsing]: <

 [Parsing]: a

 [Parsing]: href="www.google.com"

 [Parsing]: >

 [Parsing]: A end tag: </a>

 Valid syntax ...
```

2. A <a>tag with no attribute
   Eg:
   <a >

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <a > </a>

 [Parsing]: <

 [Parsing]: a

 [Parsing]: >

 [Parsing]: A end tag: </a>

 Valid syntax ...
```

3. A <a> tag can have content simple in it

   Eg:

   Google

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <a href="www.google.com">Google Link</a>

 [Parsing]: <

 [Parsing]: a

 [Parsing]: href="www.google.com"

 [Parsing]: >

 [Parsing]: Google

 [Parsing]: Link

 [Parsing]: A end tag: </a>

 Valid syntax ...
```

4. A <a> tag with different attribute values in any order.

   Eg:

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <a download="pqr"  href="www.hi.com" target="abc"> </a>

 [Parsing]: <

 [Parsing]: a

 [Parsing]: download="pqr"

 [Parsing]: href="www.hi.com"

 [Parsing]: target="abc"

 [Parsing]: >

 [Parsing]: A end tag: </a>

 Valid syntax ...
```

Test Cases for <a> tag (Invalid)

1. A <a> tag enclosing other tags
   Eg:
    hello

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

Enter the string : <a href="www.google.com"> <div>hello</div> </a>

[Parsing]: <

[Parsing]: a

[Parsing]: href="www.google.com"

[Parsing]: >

[Parsing]: <

[Parsing]: div

Invalid syntax : Error : syntax error
```

2. A <a> tag with or without attributes but invalid ending
   Eg

   a. End tag is not mached with starting tag
      Eg: <a href=”www.google.com”>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

Enter the string : <a href="www.google.com"></b>

[Parsing]: <

[Parsing]: a

[Parsing]: href="www.google.com"

[Parsing]: >

[Parsing]: </b>

Invalid syntax : Error : Unknown End Tag.
```

b. Missing endtag
   Eg: <a>

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : < a>

 [Parsing]: <

 [Parsing]: a

 [Parsing]: >

 Invalid syntax : Error : syntax error
```

Test Cases for Nested Tags (Valid)

1. A nested tag enclosing <a> tags
   Eg:
    <a>hello

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <a href="www.google.com"> <a>hello</a> </a>

 [Parsing]: <

 [Parsing]: a

 [Parsing]: href="www.google.com"

 [Parsing]: >

 [Parsing]: <

 [Parsing]: a

 [Parsing]: >

 [Parsing]: hello

 [Parsing]: A end tag: </a>

 [Parsing]: A end tag: </a>

 Valid syntax ...
```

2. A nested tag enclosing <img> tags
   Eg:


```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <a href="www.google.com"> <img src="d" alt="o" height="12"/>
 </a>

 [Parsing]: <

 [Parsing]: a

 [Parsing]: href="www.google.com"

 [Parsing]: >

 [Parsing]: <

 [Parsing]: img

 [Parsing] src="d"

 [Parsing]: alt="o"

 [Parsing]: height="12"

 [Parsing]: />

 [Parsing]: A end tag: </a>

 Valid syntax ...
```

3. A nested tag enclosing both <a> and <img> tags in any order.
   Eg:
    <a>In inner first a_tag <a> In inner second a_tag

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <a href="www.google.com"> <a>In inner first a_tag</a>  <a> I
n inner second a_tag <img src="d" alt="o" height="12"/></a> </a>

[Parsing]: <

[Parsing]: a

[Parsing]: href="www.google.com"

[Parsing]: >

[Parsing]: <

[Parsing]: a

[Parsing]: >

[Parsing]: In

[Parsing]: inner

[Parsing]: first

[Parsing]: a_tag

[Parsing]: A end tag: </a>

[Parsing]: <

[Parsing]: a

[Parsing]: >

[Parsing]: In

[Parsing]: inner

[Parsing]: second

[Parsing]: a_tag
```

```
[Parsing]: <

[Parsing]: img

[Parsing] src="d"

[Parsing]: alt="o"

[Parsing]: height="12"

[Parsing]: />

[Parsing]: A end tag: </a>

[Parsing]: A end tag: </a>

Valid syntax ...
```

Test Cases for Nested Tags (Invalid)

1. A nested tag enclosing other tags

Eg:

 hello

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

 Enter the string : <a href="www.google.com"> <div>hello</div> </a>

 [Parsing]: <

 [Parsing]: a

 [Parsing]: href="www.google.com"

 [Parsing]: >

 [Parsing]: <

 [Parsing]: div

 Invalid syntax : Error : syntax error
```

2. A nested tag enclosing other <a> or <img> tags without ending.

Eg:

 <a>hello

```
C:\Users\Divya\Desktop\Compiler_Design_Programs\Divya>a.exe

Enter the string : <a href="www.google.com"> <a>hello </a>

[Parsing]: <

[Parsing]: a

[Parsing]: href="www.google.com"

[Parsing]: >

[Parsing]: <

[Parsing]: a

[Parsing]: >

[Parsing]: hello

[Parsing]: A end tag: </a>

Invalid syntax : Error : syntax error
```

LEX Code

```
/* Lex Program for { a^n c b^n } Language. */

%{
        /*contains declaration of all the tokens in th yacc program*/
#include "y.tab.h"
        /*variables used while validation*/
int a_valid = 0,end_tag1=0;
%}

/*Regular expressions to be matched while performaing lexical analysis*/

/*RE for whitespaces*/
Ws              ([ \t    ]+)

/*RE for letters*/
Letter          [A-Za-z]

/*RE for digits*/
Digit           [0-9]

/*RE for identifiers*/
Name                    {Letter}({Letter}|{Digit}|[_])*

/*RE for content inside <a> tag*/
Content         {Name}|{Letter}|{Digit}

/*attributes of <a> tag*/
AAttrList
        "download"|"href"|"hreflag"|"media"|"ping"|"referrerpolicy"|"rel"|"target"|"type"

/*attribute list for <a> tag*/
AAttr                           {AAttrList}

/*attributes of <img> tag*/
ImgAttList              "height"|"width"|"crossorigin"|"use-
credentials"|"ismap"|"loading"|"longdesc"|"referrerpolicy"|"no-referrer-when-
downgrade"|"origin"|"origin-when-cross-origin"|"unsafe-url"|"sizes"|"usemap"

/*attribute list for image tag*/
ImgOtherAtt     {ImgAttList}
```

```
/*RE for attribute value*/
AttValue       (\"[^<&"]*\")|(\'[^<&']*\')

        //end of declaration section

%%
        /*translation rules
        pattern {
                                action
                        }
        */

        /* "<"" will be matched here.*/
[<] {   printf("\n [Parsing]: %s\n", yytext);
                return START_ANGLE_BRACKET;}

        /* "a" will be matched here.*/
[a] {           /*a_valid is used for validating content of <a> tag.*/
                a_valid =1;
                printf("\n [Parsing]: %s\n", yytext);
                return A_TAG;}

"img" { printf("\n [Parsing]: %s\n", yytext);
                return IMG_TAG;}

{Ws} {;}

        /*anchor tag's attribute is matched here*/
{AAttrList}{Ws}?[=]{Ws}?{AttValue}{Ws}? {
                                                        //valid attribute for a tag
                                                        printf("\n [Parsing]:
%s\n", yytext);
                                                        return A_ATTRIBUTE;
                                                }

[>] {   /*end_tag1 is used for validating content of <a> tag.*/
                end_tag1=1;
                printf("\n [Parsing]: %s\n", yytext);
                return END_TAG1;}

                /*End tag for </a> is matched here*/
[<]{Ws}?[/]{Ws}?[a]{Ws}?[>]{
```

```
                                        //valid endtag
                                        printf("\n [Parsing]: A end tag: %s\n",
yytext);
                                        return A_END_TAG;

                                }

        /*Unlnown End tags matched here*/
[<]{Ws}?[/]{Ws}?{Name}{Ws}?[>]  {
                                        //valid endtag
                                        printf("\n [Parsing]: %s\n", yytext);
                                        yyerror("Unknown End Tag.");

                                }


    /*content inside <a> tag is matched here.*/
{Content}    {
                        if(a_valid==1 && end_tag1==1)
                        {
                                printf("\n [Parsing]: %s\n", yytext);
                                return A_CONTENT;
                        }
                        else
                        {
                                printf("\n [Parsing]: %s\n", yytext);
                                yyerror("Unknown text.");
                        }

                }

    /*attribute src of image tag with attribute value is matched here.*/
"src"{Ws}?[=]{Ws}?{AttValue}{Ws}?       {

                                        printf("\n [Parsing] %s\n", yytext);
                                        return IMG_SRC;

                                }

    /*attribute alt of image tag with attribute value is matched here.*/
"alt"{Ws}?[=]{Ws}?{AttValue}{Ws}?       {

                                        printf("\n [Parsing]: %s\n", yytext);
```

```
                                                    return IMG_ALT;

                                                }

        /*all other valid img attributes is matched here*/
{ImgOtherAtt}{Ws}?[=]{Ws}?{AttValue}{Ws}?       {
                                                    //valid attribute for img
tag
                                                    printf("\n [Parsing]:
%s\n", yytext);
                                                    return
IMG_OTHER_ATTRIBUTE;
                                                }

        /*all other attributes which are invalid are matched here*/
{Name}{Ws}?[=]{Ws}?{AttValue}{Ws}?  {
                                                    //attribute list doesnot content this
attribute
                                                    printf("\n [Parsing]: %s\n", yytext);
                                                    yyerror("Unknown attribute.");
                                                }

        /*End tag for image "/>" is matched here*/
[/]{Ws}?[>]{Ws}?{
                    //valid endtag for img
                    printf("\n [Parsing]: %s\n", yytext);
                    return IMG_END_TAG2;
                }


        /*End tag for image "></img> is matched here*/
[>]{Ws}?[<]{Ws}?[/]{Ws}?("img"){Ws}?[>]{Ws}?    {
                            //valid endtag
                            printf("\n [Parsing]: Image end tag: %s\n", yytext);
                            return IMG_END_TAG3;

                    }

        /*newline character*/
[\n] { return NL; }

        /*Everything else is matched here.*/
. { printf("\n [Parsing]: %s\n", yytext);
```

```
        yyerror("Unknown text.");}

%%

int yywrap()
{
        return 1;
}
```

YACC Code

```
/*
Y File
Implementing parser using lex and yacc to validate <img> and <a> tag of html
*/

%{
#include<stdio.h>
#include<stdlib.h>
int yylex();
int yyerror();
%}

        //tokens generated by lexical analyzer and passed to parser.
%token START_ANGLE_BRACKET NL A_TAG A_ATTRIBUTE SYNTAX_ERROR
EQUAL ATTR_VALUE END_TAG1 A_CONTENT A_END_TAG IMG_TAG IMG_ALT
IMG_SRC IMG_OTHER_ATTRIBUTE IMG_END_TAG2 IMG_END_TAG3

%%

        //Grammer rules.
        //if stmt is reached msg valid syntax is printed.

stmt : S NL { printf("\n Valid syntax ...\n\n");
          exit(0);}
                ;


        //start symbol(S) goes to <img> tag or <a> tag.
        // START_ANGLE_BRACKET = "<".

S       : START_ANGLE_BRACKET A_TAG anchor_attrs
        | START_ANGLE_BRACKET IMG_TAG img_comp_attr img_other_attr img_ending
        ;


img_comp_attr      : img_comp_attr_1
                        | img_comp_attr_2
                        ;
```

```
img_comp_attr_1 : IMG_SRC IMG_ALT
                ;
img_comp_attr_2 : IMG_ALT IMG_SRC
                ;
img_other_attr  : IMG_OTHER_ATTRIBUTE img_other_attr
                |
                ;
        //img can have three type of end tag >, /> or </img>.
img_ending      : END_TAG1
                | IMG_END_TAG2
                | IMG_END_TAG3
                ;


anchor_attrs    : A_ATTRIBUTE anchor_attrs
                | END_TAG1 anchor_rem A_END_TAG
                ;

        //Tags nesting is done here in <a> tag.
anchor_rem      : content S anchor_rem
                | content
                ;
content         : A_CONTENT content
                |
                ;

%%

void main()
{
  printf("\n Enter the string : ");
  yyparse();
}

int yyerror(char *msg)
{
  printf("\n Invalid syntax : Error : %s\n\n",msg);
  exit(0);
}
```

References

Compilers, principles, techniques, and tools / Alfred V. Aho, Ravi. Sethi, Jeffrey D. Ullman. 1986. ISBN0-321-48681-1

https://www.geeksforgeeks.org/ for tag syntax

THANK YOU