{JSON}

MODIFIER

Members:

Arushi Agarwal - 13503896

Amit Bansal - 13503885

Shubham Jain - 13503883

Batch-B13

**Introduction:**

**What is JSON?**

JavaScript Object Notation (JSON) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects.

Despite its relationship to JavaScript, it is language-independent, with parsers available for most programming languages.

**What is XML?**

Extensible Markup Language (XML) is a set of rules for encoding documents in

machine-readable form. XML’s design goals emphasize simplicity, generality, and

usability over the Internet.

**Objective:**

Our main objective is to make a tool which can convert JSON to other different types of data representation languages like XML, CSV, HTML and can also beautify and minify the json data.

**What can you do with JSON Modifier?**

1. **It helps to convert JSON into XML format.**
2. **It helps to convert JSON into HTML table structure.**
3. **It helps to convert JSON into CSV format.**
4. **Validates JSON.**
5. **Minify the code.**
6. **Formats and Beautify the code to proper indentation for better understanding.**

**JSON to XML**

**Grammar for converting json to xml with semantic rule for each production**

P -> object

*{print(object.val)}*

object -> {ob}

*{object.val=ob.val}*

ob -> A,ob1

*{ob.val=A.val||ob1.val}*

ob -> A

*{ob.val=A.val}*

A -> N:V

*{A.val=V.val, V.name=N.val}*

N -> str

*{N.val=str.val}*

V -> str

*{V.val=<V.name>str.val</V.name>}*

V -> num

*{V.val=<V.name>num.val</V.name>}*

V -> bool

*{V.val=<V.name>bool.val</V.name>}*

V -> float

*{V.val=<V.name>float.val</V.name>}*

V -> array

*{array.name=V.name, V.val=array.val}*

V -> object

*{V.val=<V.name>object.val</V.name>}*

str -> "s"

*{str.val=s.val}*

array -> [arr]

*{arr.name=array.name, array.val=arr.val}*

arr -> V,arr1

*{arr1.name=arr.name, V.name=arr.name, arr.val=V.val||arr1.val}*

arr -> V

*{arr.val=V.val, V.name=arr.name}*

float -> num.num1

*{float.val=num.val||'.'||num1.val}*

num -> digit num1

*{num.val=digit.val||num1.val}*

num -> digit

*{num.val=digit.val}*

bool -> true

*{bool.val=true}*

bool -> false

*{bool.val=false}*

digit -> 0

*{digit.val=0}*

|

|

|

|

digit -> 9

*{digit.val=9}*

s -> alnum s1

*{s.val=alnum.val||s1.val}*

s -> alnum

*{s.val=alnum.val}*

alnum -> a

*{alnum.val=a}*

|

|

|

|

alnum -> z

*{alnum.val=z}*

alnum -> A

*{alnum.val=A}*

|

|

|

|

alnum -> Z

*{alnum.val=Z}*

alnum -> 0

*{alnum.val=0}*

|

|

|

|

alnum -> 9

*{alnum.val=9}*

alnum -> \"

*{alnum.val="}*

alnum -> '

*{alnum.val='}*

alnum -> \\

*{alnum.val=\}*

**Parse tree for the following json:**

{

"shubham": {

"abc": 20,

"sh": [

"hello",

{

"fn": "use"

}

],

"hi": {

"we": "you",

"yes": 10.35

}

},

"yups": [

{

"yo": "why"

},

20,

"yeah"

]

}

parsetree.png

**The above json will generate the following xml data:**

<shubham>

<abc>20</abc>

<sh>hello</sh>

<sh>

<fn>use</fn>

</sh>

<hi>

<we>you</we>

<yes>10.35</yes>

</hi>

</shubham>

<yups>

<yo>why</yo>

</yups>

<yups>20</yups>

<yups>yeah</yups>

**JSON to CSV**

**Grammar for converting json to csv with semantic rule for each production**

int arrindex=0;

P -> object

*{print(object.val)}*

object -> {ob}

*{object.val=ob.val}*

ob -> A,ob1

*{ob.val=A.val||ob1.val}*

ob -> A

*{ob.val=A.val}*

A -> N:V

*{*

*If(V.type==4) then*

*A.val=N.val||”.”||V.val*

*Elseif(V.type==1) then*

*A.val=N.val||”:”||V.val*

*}*

N -> str

*{N.val=str.val}*

V -> str

*{ V.val=str.val*

*V.type=1*

*}*

V -> num

*{ V.val=num.val*

V.type=1

}

V -> bool

*{ V.val=bool.val*

V.type=1

}

V -> float

*{ V.val=float.val*

V.type=1

}

V -> array

*{ V.val=array.val*

V.type=4

}

V -> object

*{ V.val=object.val*

V.type=4

}

str -> "s"

*{str.val=s.val}*

array -> [arr]

*{array.val=arr.val*

*arrindex=0}*

arr -> V,arr1

*{*

*If(V.type==4) then*

*arr.val=arrindex||”.”||V.val||*arr1.val

*Elseif(V.type==1) then*

*arr.val=arrindex||”:”||V.val||*arr1*.val*

*arrindex++*

*}*

arr -> V

*{*

*If(V.type==4) then*

*arr.val=arrindex||”.”||V.val*

*Elseif(V.type==1) then*

*arr.val=arrindex||”:”||V.val*

*arrindex++*

*}*

float -> num.num1

*{float.val=num.val||'.'||num1.val}*

num -> digit num1

*{num.val=digit.val||num1.val}*

num -> digit

*{num.val=digit.val}*

bool -> true

*{bool.val=true}*

bool -> false

*{bool.val=false}*

digit -> 0

*{digit.val=0}*

|

|

|

|

digit -> 9

*{digit.val=9}*

s -> alnum s1

*{s.val=alnum.val||s1.val}*

s -> alnum

*{s.val=alnum.val}*

alnum -> a

*{alnum.val=a}*

|

|

|

|

alnum -> z

*{alnum.val=z}*

alnum -> A

*{alnum.val=A}*

|

|

|

|

alnum -> Z

*{alnum.val=Z}*

alnum -> 0

*{alnum.val=0}*

|

|

|

|

alnum -> 9

*{alnum.val=9}*

alnum -> \"

*{alnum.val="}*

alnum -> '

*{alnum.val='}*

alnum -> \\

*{alnum.val=\}*

**Parse tree for the following json:**

{

"menu": {

"id": "file",

"value": ["ddd",13]

}

}

C:\Users\mrb\Downloads\Untitled Diagram.png

**The above json will generate the following csv data:**

menu.id : file

menu.value.1 : ddd

menu.value.0 : 13

**JSON to HTML**

**Grammar for converting json to html with semantic rule for each production**

P -> object

*{*

*P.val=”<html><head><style>table,th,td {border: 1px solid black;border-collapse: collapse;}*

*th,td {padding: 5px;}</style></head><body>”||object.val||”</body></html>”*

*}*

object -> {ob}

*{object.val=”<table>”||ob.val||”</table>”}*

ob -> A,ob1

*{ob.val=A.val||ob1.val}*

ob -> A

*{ob.val=A.val}*

A -> N:V

*{*

*A.val=”<tr><td>”||N.val||”</td><td>”||V.val||”</td></tr>”*

*}*

N -> str

*{N.val=str.val}*

V -> str

*{ V.val=str.val*

*}*

V -> num

*{ V.val=num.val*

}

V -> bool

*{ V.val=bool.val*

}

V -> float

*{ V.val=float.val*

}

V -> array

*{ V.val=array.val*

}

V -> object

*{ V.val=object.val*

}

str -> "s"

*{str.val=s.val}*

array -> [arr]

*{array.val=”<table>”||arr.val||”</table>”}*

arr -> V,arr1

*{*

*arr.val=”<tr><td>”||V.val||”</td></tr>”||*arr1*.val*

*}*

arr -> V

*{*

*arr.val=”<tr><td>”||V.val||”</td></tr>”*

*}*

float -> num.num1

*{float.val=num.val||'.'||num1.val}*

num -> digit num1

*{num.val=digit.val||num1.val}*

num -> digit

*{num.val=digit.val}*

bool -> true

*{bool.val=true}*

bool -> false

*{bool.val=false}*

digit -> 0

*{digit.val=0}*

|

|

|

|

digit -> 9

*{digit.val=9}*

s -> alnum s1

*{s.val=alnum.val||s1.val}*

s -> alnum

*{s.val=alnum.val}*

alnum -> a

*{alnum.val=a}*

|

|

|

|

alnum -> z

*{alnum.val=z}*

alnum -> A

*{alnum.val=A}*

|

|

|

|

alnum -> Z

*{alnum.val=Z}*

alnum -> 0

*{alnum.val=0}*

|

|

|

|

alnum -> 9

*{alnum.val=9}*

alnum -> \"

*{alnum.val="}*

alnum -> '

*{alnum.val='}*

alnum -> \\

*{alnum.val=\}*

**Consider the following JSON:**

{

"menu": {

"id": "file",

"value": ["ddd",13,10.3],

"popup": 18

},

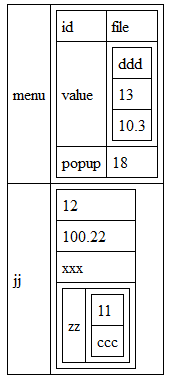
"jj":[12,100.22,"xxx",{"zz":[11,"ccc"]}]

}

**The above json will generate the following html data:**

<html><head><style>table,th,td {border: 1px solid black;border-collapse: collapse;} th,td {padding: 5px;}</style></head><body><table><tr><td>menu</td><td><table><tr><td>id</td><td>file</td></tr><tr><td>value</td><td><table><tr><td>ddd</td></tr><tr><td>13</td></tr><tr><td>10.3</td></tr></table></td></tr><tr><td>popup</td><td>18</td></tr></table></td></tr><tr><td>jj</td><td><table><tr><td>12</td></tr><tr><td>100.22</td></tr><tr><td>xxx</td></tr><tr><td><table><tr><td>zz</td><td><table><tr><td>11</td></tr><tr><td>ccc</td></tr></table></td></tr></table></td></tr></table></td></tr></table></body></html>

**On opening the above html code using the browser, it shows the following output:**

**­­­**