



Basic SemiXML tests

Testing the basic use of SemiXML

Introduction

SemiXML is a library which translates an equally named language into XML. There are several syntactic constructs which look very much the same as the XML language.

We need to write a convenience routine to call the parser several times from the tests. This way we keep the tests simple.

```

use SemiXML;

sub parse ( Str $content --> Str ) {

    state SemiXML::Sxml $x .= new;
    $x.parse(:$content);
    ~$x;
}

```

Plain top level elements

First a few tests to generate top level elements

```

my $xml = parse('$st []');
is $xml, '<st/>', 'T0';
$xml = parse('$st [ abc ]');
is $xml, '<st>abc</st>', 'T1';

```

- ✓ **T0:** The element generated is `<st/>`.
- ✓ **T1:** The element generated is `<st>abc</st>`.

```

$xml = parse(Q:q@$st a1=w a2='g g' a3="h h" [ ]@);
like $xml, /'a1="w"/, 'T2';
like $xml, /'a2="g g"/, 'T3';
like $xml, /'a3="h h"/, 'T4';
dies-ok { $xml = parse('$st [ $f w [ ] hj ]'); }, 'T5';

```

- ✓ **T2:** `a1="w"` attribute
- ✓ **T3:** `a2="g g"` attribute
- ✓ **T4:** `a3="h h"` attribute
- ✓ **T5:** Elementary parse failures are trapped. Here the first attribute `w` to element `f` is unfinished. Unfortunately the location where the error exactly happens is not always accurate.

Basic nesting

Nesting is done by specifying other tags in the body of a tag.

```
$xml = parse('$t1 [ $t2 [] $t3[]]');  
is $xml, '<t1><t2/><t3/></t1>', 'T6';
```

✓ **T6:** t2 and t3 are the nested tags

```
$xml = parse('$t1 [ $**t2 [] $t3[]]');  
is $xml, '<t1> <t2/> <t3/></t1>', 'T7';  
$xml = parse('$t1 [ $|*t2 [] $t3[]]');  
is $xml, '<t1><t2/> <t3/></t1>', 'T8';  
$xml = parse('$t1 [ $*|t2 [] $t3[]]');  
is $xml, '<t1> <t2/><t3/></t1>', 'T9';
```

✓ **T7:** nested tags with spaces around element t2 using \$**

✓ **T8:** nested tags with a space on the right of t2 with \$|*

✓ **T9:** nested tags with a space on the left of t2 done with \$*|

Generated using SemiXML, SxmlLib::Testing::TestDoc, XML