

python / cpython Public

<> Code Issues 5k+ Pull requests 2.2k Actions Projects Security and q

Commit ab0893f



serhiy-storchaka authored on Jul 3, 2025 · 76 / 78 · Verified

[3.12] [gh-135462](#): Fix quadratic complexity in processing special input in HTMLParser ([GH-135464](#)) ([GH-135483](#))

End-of-file errors are now handled according to the HTML5 specs -- comments and declarations are automatically closed, tags are ignored. (cherry picked from commit [6eb6c5d](#))

3.12 (#135483) · v3.12.13 v3.12.12

1 parent [033aa5c](#) commit [ab0893f](#)

3 files changed +116 -23 lines changed

Top

Filter files...

- Lib
 - html
 - parser.py
 - test
 - test_htmlparser.py
- Misc/NEWS.d/next/Security
 - 2025-06-13-15-55-22.gh-issue-135462.KBeJpc.rst

3 files changed +116 -23 lines changed

Search within code

Lib/html/parser.py

```

↑... @@ -25,6 +25,7 @@
25 25 charref = re.compile('&#(?:[0-9]+|[xX][0-9a-fA-F+)][^0-9a-fA-F]')
26 26
27 27 starttagopen = re.compile('<[a-zA-Z]')
```

```

28 + endtagopen = re.compile('</[a-zA-Z]')
28 29 piclose = re.compile('>')
29 30 commentclose = re.compile(r'--\s*>')
30 31 # Note:
@@ -177,25 +178,43 @@ def goahead(self, end):
177 178         k = self.parse_pi(i)
178 179         elif startswith("<", i):
179 180         k = self.parse_html_declaration(i)
180 -         elif (i + 1) < n:
181 +         elif (i + 1) < n or end:
181 182             self.handle_data("<")
182 183             k = i + 1
183 184         else:
184 185             break
185 186         if k < 0:
186 187             if not end:
187 188                 break
188 -         k = rawdata.find('>', i + 1)
189 -         if k < 0:
190 -             k = rawdata.find('<', i + 1)
191 -             if k < 0:
192 -                 k = i + 1
193 -             else:
194 -                 k += 1
195 -             if self.convert_charrefs and not self.cdata_elem:
196 -                 self.handle_data(unescape(rawdata[i:k]))
189 +         if starttagopen.match(rawdata, i): # < + letter
190 +             pass
191 +         elif startswith("</", i):
192 +             if i + 2 == n:
193 +                 self.handle_data("</")
194 +             elif endtagopen.match(rawdata, i): # </ + letter
195 +                 pass
196 +             else:
197 +                 # bogus comment
198 +                 self.handle_comment(rawdata[i+2:])
199 +         elif startswith("<!--", i):
200 +             j = n
201 +             for suffix in ("--!", "--", "-"):

```

```

202 +         if rawdata.endswith(suffix, i+4):
203 +             j -= len(suffix)
204 +             break
205 +         self.handle_comment(rawdata[i+4:j])
206 +     elif startswith("<![CDATA[", i):
207 +         self.unknown_decl(rawdata[i+3:])
208 +     elif rawdata[i:i+9].lower() == '<!doctype':
209 +         self.handle_decl(rawdata[i+2:])
210 +     elif startswith("<!", i):
211 +         # bogus comment
212 +         self.handle_comment(rawdata[i+2:])
213 +     elif startswith("<?", i):
214 +         self.handle_pi(rawdata[i+2:])
197 215         else:
198 -         self.handle_data(rawdata[i:k])
216 +         raise AssertionError("we should not get here!")
217 +         k = n
199 218         i = self.updatepos(i, k)
200 219     elif startswith("&#", i):
201 220         match = charref.match(rawdata, i)

```

Lib/test/test_htmlparser.py

```

@@ -5,6 +5,7 @@
5 5     import unittest
6 6
7 7     from unittest.mock import patch
8 + from test import support
8 9
9 10
10 11     class EventCollector(html.parser.HTMLParser):
@@ -393,28 +394,34 @@ def test_tolerant_parsing(self):
393 394         ('data', '<'),
394 395         ('starttag', 'bc<', [('a', None)]),
395 396         ('endtag', 'html'),
396 -         ('data', '\n", [('data', '<>')])
402 +     self._run_check("< >", [('data', '< >')])
403 +     self._run_check("< ", [('data', '< ')])
401 404     self._run_check("</>", [])
405 +     self._run_check("<$>", [('data', '<$>')])
402 406     self._run_check("</$>", [('comment', '$')])
403 407     self._run_check("</", [('data', '</')])
404 -     self._run_check("</a", [('data', '</a')])
408 +     self._run_check("</a", [])
409 +     self._run_check("</ a>", [('endtag', 'a')])
410 +     self._run_check("</ a", [('comment', ' a')])
405 411     self._run_check("<a<a>", [('starttag', 'a<a', [ ])])
406 412     self._run_check("</a<a>", [('endtag', 'a<a')])
407 -     self._run_check("<!", [('data', '<!')])
408 -     self._run_check("<a", [('data', '<a')])
409 -     self._run_check("<a foo='bar'", [('data', "<a foo='bar'")])
410 -     self._run_check("<a foo='bar", [('data', "<a foo='bar'")])
411 -     self._run_check("<a foo='>", [('data', "<a foo='>")])
412 -     self._run_check("<a foo='>", [('data', "<a foo='>")])
413 +     self._run_check("<!", [('comment', '!')])
414 +     self._run_check("<a", [])
415 +     self._run_check("<a foo='bar'", [])
416 +     self._run_check("<a foo='bar", [])
417 +     self._run_check("<a foo='>", [])
418 +     self._run_check("<a foo='>", [])
413 419     self._run_check("<a$b>", [('starttag', 'a$b', [ ])])
414 420     self._run_check("<a$b>", [('starttag', 'a$b', [ ])])
415 421     self._run_check("<a$b/>", [('startendtag', 'a$b', [ ])])
416 422     self._run_check("<a$b >", [('starttag', 'a$b', [ ])])
417 423     self._run_check("<a$b />", [('startendtag', 'a$b', [ ])])
424 +     self._run_check("</a$b>", [('endtag', 'a$b')])
418 425
419 426     def test_slashes_in_starttag(self):
420 427         self._run_check('<a foo="var"/>', [('startendtag', 'a', [('foo',
            'var')])])

```



```
@@ -539,13 +546,56 @@ def test_EOF_in_charref(self):
```

```
539 546         for html, expected in data:
540 547             self._run_check(html, expected)
541 548
542 -     def test_broken_comments(self):
543 -         html = ('<! not really a comment >')
549 +     def test_eof_in_comments(self):
550 +         data = [
551 +             ('<!--', [('comment', '')]),
552 +             ('<!---', [('comment', '')]),
553 +             ('<!---', [('comment', '')]),
554 +             ('<!---', [('comment', '-')]),
555 +             ('<!---', [('comment', '--')]),
556 +             ('<!--!', [('comment', '')]),
557 +             ('<!--!', [('comment', '-!')]),
558 +             ('<!--!>', [('comment', '-!>')]),
559 +             ('<!--foo', [('comment', 'foo')]),
560 +             ('<!--foo-', [('comment', 'foo')]),
561 +             ('<!--foo--', [('comment', 'foo')]),
562 +             ('<!--foo--!', [('comment', 'foo')]),
563 +             ('<!--<!--', [('comment', '<!')]),
564 +             ('<!--<!--!', [('comment', '<!')]),
565 +         ]
566 +         for html, expected in data:
567 +             self._run_check(html, expected)
568 +
569 +     def test_eof_in_declarations(self):
570 +         data = [
571 +             ('<!', [('comment', '')]),
572 +             ('<!--', [('comment', '-')]),
573 +             ('<![', [('comment', '[')]),
574 +             ('<![CDATA[', [('unknown decl', 'CDATA[')]),
575 +             ('<![CDATA[x', [('unknown decl', 'CDATA[x')]),
576 +             ('<![CDATA[x]', [('unknown decl', 'CDATA[x')]),
577 +             ('<![CDATA[x]]', [('unknown decl', 'CDATA[x]]')]),
578 +             ('<!DOCTYPE', [('decl', 'DOCTYPE')]),
579 +             ('<!DOCTYPE ', [('decl', 'DOCTYPE ')]),
580 +             ('<!DOCTYPE html', [('decl', 'DOCTYPE html')]),
581 +             ('<!DOCTYPE html ', [('decl', 'DOCTYPE html ')]),
582 +             ('<!DOCTYPE html PUBLIC', [('decl', 'DOCTYPE html PUBLIC')]),
```

```

583 +         ('<!DOCTYPE html PUBLIC "foo', [('decl', 'DOCTYPE html PUBLIC
      "foo')])),
584 +         ('<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "foo',
585 +         [('decl', 'DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
      "foo')])),
586 +     ]
587 +     for html, expected in data:
588 +         self._run_check(html, expected)
589 +
590 +     def test_bogus_comments(self):
591 +         html = ('<!ELEMENT br EMPTY>'
592 +                '<! not really a comment >'
593 +                '<! not a comment either -->'
594 +                '<! -- close enough -->'
595 +                '<!><!-- this was an empty comment>'
596 +                '<!!! another bogus comment !!!>')
597 +         expected = [
598 +             ('comment', 'ELEMENT br EMPTY'),
599 +             ('comment', ' not really a comment '),
600 +             ('comment', ' not a comment either --'),
601 +             ('comment', ' -- close enough --'),
602 +
603 +             @@ -600,6 +650,26 @@ def test_convert_charrefs_dropped_text(self):
604 +
605 +             ('endtag', 'a'), ('data', ' bar & baz')]
606 +         )
607 +
608 +     @support.requires_resource('cpu')
609 +     def test_eof_no_quadratic_complexity(self):
610 +         # Each of these examples used to take about an hour.
611 +         # Now they take a fraction of a second.
612 +         def check(source):
613 +             parser = html.parser.HTMLParser()
614 +             parser.feed(source)
615 +             parser.close()
616 +
617 +             n = 120_000
618 +             check("<a " * n)
619 +             check("<a a=" * n)
620 +             check("</a " * 14 * n)
621 +             check("</a a=" * 11 * n)
622 +             check("<!-- " * 4 * n)

```

```
667 +         check("<!" * 60 * n)
668 +         check("<?" * 19 * n)
669 +         check("</$" * 15 * n)
670 +         check("<![CDATA[" * 9 * n)
671 +         check("<!doctype" * 35 * n)
672 +
```

```
603 673
604 674     class AttributesTestCase(TestCaseBase):
605 675
```



...5-06-13-15-55-22.gh-issue-135462.KBeJpc.rst



```
... @@ -0,0 +1,4 @@
1 + Fix quadratic complexity in processing specially crafted input in
2 + :class:`html.parser.HTMLParser`. End-of-file errors are now handled according
3 + to the HTML5 specs -- comments and declarations are automatically closed,
4 + tags are ignored.
```

Comments 0