

---

# **Rattle.py Documentation**

***Release 0.0.2a1***

**Frodo821**

**Apr 20, 2019**



---

## Contents:

---

<b>1</b>	<b>rattlepy : Rattle.py API</b>	<b>1</b>
<b>2</b>	<b>rattlepy.elements : Rattle.py API</b>	<b>3</b>
<b>3</b>	<b>rattlepy.templating : Rattle.py API</b>	<b>5</b>
<b>4</b>	<b>rattlepy.utils : Rattle.py API</b>	<b>7</b>
<b>5</b>	<b>Module rattlepy.environment : Rattle.py API Reference</b>	<b>9</b>
<b>6</b>	<b>Rattle.py API</b>	<b>11</b>
<b>7</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>



# CHAPTER 1

---

## rattlepy : Rattle.py API

---

- : *Elements*

Rattle.py - A Pure Python Templating Library for HTML A pure python templating library for html. Rattle.py has no special notation like Django or Jinja. For example:

```
<html>
  <head>
    <title>Hello, PTL!</title>
  </head>
  <body>
    <h1 class="heading">Hello, PTL!</h1>
  </body>
</html>
```

The above HTML equals to below Python code with rattle.py:

```
greeting = "Hello, PTL!"
with html() as html:
    with head():
        with title():
            text(greeting)
    with body():
        with node("h1", className="heading"):
            text(greeting)

# show as HTML
print(html)
```

And then, you can also make reusable components by yourself:

```
def greet(name):
    with node("div", className="greet-wrapper") as component:
        with node("h1"):
            text(f"Hello, {name}=san")
```

(continues on next page)

(continued from previous page)

```
    with node("button", className="ok-btn"):
        text("ok!")
    return component

# and using:
with greet("User"): pass
```

Enjoy!

- : *Elements*

---

### rattlepy.elements : Rattle.py API

---

- : *Utils*
- : *Rattlepy*

HTML element short-handing functions

```
rattlepy.elements.a(**kwargs)
    return Element("a", attributes...)

rattlepy.elements.article(**kwargs)
    article return Element("article", attributes... )

rattlepy.elements.body(**kwargs)
    body return Element("body", attributes...)

rattlepy.elements.div(**kwargs)
    div return Element("div", attributes...)

rattlepy.elements.footer(**kwargs)
    footer return Element("footer", attributes...)

rattlepy.elements.h1(**kwargs)
    h1 return Element("h1", attributes...)

rattlepy.elements.h2(**kwargs)
    h2 return Element("h2", attributes...)

rattlepy.elements.h3(**kwargs)
    h3 return Element("h3", attributes...)

rattlepy.elements.h4(**kwargs)
    h4 return Element("h4", attributes...)

rattlepy.elements.h5(**kwargs)
    h5 return Element("h5", attributes...)

rattlepy.elements.h6(**kwargs)
    h6 return Element("h6", attributes...)
```

```
rattlepy.elements.head(**kwargs)
    head return Element("head", attributes...)

rattlepy.elements.header(**kwargs)
    header return Element("header", attributes...)

rattlepy.elements.hr(**kwargs)
    hr return Element("hr", attributes...)

rattlepy.elements.html(**kwargs)
    html return Element("html", attributes...)

rattlepy.elements.img(**kwargs)
    img return Element("img", attributes...)

rattlepy.elements.li(**kwargs)
    li return Element("li", attributes...)

rattlepy.elements.link(**kwargs)
    link return Element("link", attributes...)

rattlepy.elements.main(**kwargs)
    main return Element("main", attributes...)

rattlepy.elements.meta(**kwargs)
    meta return Element("meta", attributes...)

rattlepy.elements.ol(**kwargs)
    ol return Element("ol", attributes...)

rattlepy.elements.p(**kwargs)
    paragraph return Element("p", attributes...)

rattlepy.elements.script(**kwargs)
    script return Element("script", attributes...)

rattlepy.elements.span(**kwargs)
    span return Element("span", attributes...)

rattlepy.elements.style(**kwargs)
    style return Element("style", attributes...)

rattlepy.elements.title(**kwargs)
    title return Element("title", attributes...)

rattlepy.elements.ul(**kwargs)
    ul return Element("ul", attributes...)

rattlepy.elements.setTitle(string)
    :
```

```
with title():
    text(string)
```

with:

```
with setTitle("HogeHoge Page"): pass
```

- : *Utils*
- : *Rattlepy*



## rattlepy.templating : Rattle.py API

- : *Elements*
- : *Templating*

rattlepy.templating.**escapeHtmlEntities** (*string*)

**class** rattlepy.templating.**Element** (*tag*, \*, *className=None*, *\*\*kwargs*)  
:

```
with Element (tagname, attributes...):  
    ...
```

”class””className”

```
with Element (tagname, **{ 'class': 'my-class' }):  
    ...
```

Python”data-“

**exposes** (*element=None*)  
:

```
with Element ("hoge") as hoge:  
    # :code:`hoge`  
    with Element ("some-inner") as inner:  
        hoge.exposes (inner)  
  
with hoge:  
    # :code:`some-inner`  
    with Element ("other-element"):  
        ...  
    hoge.exposes ()  
  
with hoge:  
    # :code:`hoge`
```

(continues on next page)

(continued from previous page)

```
with Element("some-other-element"):  
    ...
```

```
serialize(formatter='human_friendly',force_add_doctype=False)  
HTML str(elem)
```

```
formatter["human_friendly", "minify"]human_friendly" force_add_doctype=Truedoctype
```

```
class rattlepy.templating.SelfClosedElement(tag, *, _outer=2, className=None,  
                                              **kwargs)  
:
```

```
with Element("hoge"):  
    SelfClosedElement(tagname, attributes...)
```

```
addself(*, outer=1)  
:
```

```
with some_parent_node:  
    # some_parent_node  
    SelfClosedElement('hr').addself()
```

```
rattlepy.templating.text(content)  
contentstr
```

```
:  
  
with Element("hoge"):  
    text(''\n|some  
|multiline  
|text'')
```

```
|
```

```
& HTML rtext
```

```
rattlepy.templating.node(tag, **kwargs)  
Element(tag, attributes...)
```

```
rattlepy.templating.closed(tag, **kwargs)  
SelfClosedElement(tag, attributes...)
```

- : *Elements*
- : *Templating*

## CHAPTER 4

---

### rattlepy.utils : Rattle.py API

---

- : *Templating*
- : *Elements*

Utility functions for making html more easily.

`rattlepy.utils.createHeader` (*title*, \**metas*)

Create head element. Usage:

```
with createHeader(  
    "Page Title",  
    {"charset": "utf-8"}):  
    ...
```

This function equals to the code:

```
with head():  
    for m in [{"charset": "utf-8"}]:  
        meta(**m)  
    setTitle("Page Title")
```

`rattlepy.utils.scaffold` (*header*: *rattlepy.templating.Element*)

Create html scaffold. This feature is under experimental.

- : *Templating*
- : *Elements*



---

## Module `rattlepy.environment` : Rattle.py API Reference

---

- **Previous document:** *Templating*

Placeholder variables implementation

**class** `rattlepy.environment.Environment`

Holding values to replace placeholders. Threads have a separate set of variables.

THIS IS AN EXPERIMENTAL FEATURE.

Usage:

```
env = Environment()

with scaffold(createHeader("Page Title")) as html:
    with h1():
        # define a placeholder named 'title'
        text(env.define('title'))

env.title = 'Test Title'
# or
# env['title'] = 'Test Title'

print(html)

# finalizes on a certain thread.
env.dispose()
```

**define** (*name*)

Create a placeholder.

**dispose** ()

Delete all data on the certain thread.

Placeholder class

**class** `rattlepy.environment.placeholder.Placeholder` (*parent, name*)

A class for placeholder in html node trees.

THIS IS AN EXPERIMENTAL FEATURE.

- **Previous document:** [\*Templating\*](#)

## CHAPTER 6

---

### Rattle.py API

---

#### Modules:

- *rattlepy*
- *rattlepy.elements*
- *rattlepy.templating*
- *rattlepy.utils*
- *rattlepy.utils*





## CHAPTER 7

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`
- *Rattle.py API*



### **r**

- `rattlepy`, [1](#)
- `rattlepy.elements`, [3](#)
- `rattlepy.environment`, [9](#)
- `rattlepy.environment.placeholder`, [9](#)
- `rattlepy.templating`, [5](#)
- `rattlepy.utils`, [7](#)



**A**

`a()` (in module `rattlepy.elements`), 3  
`addself()` (`rattlepy.templating.SelfClosedElement`  
method), 6  
`article()` (in module `rattlepy.elements`), 3

**B**

`body()` (in module `rattlepy.elements`), 3

**C**

`closed()` (in module `rattlepy.templating`), 6  
`createHeader()` (in module `rattlepy.utils`), 7

**D**

`define()` (`rattlepy.environment.Environment` method),  
9  
`dispose()` (`rattlepy.environment.Environment`  
method), 9  
`div()` (in module `rattlepy.elements`), 3

**E**

`Element` (class in `rattlepy.templating`), 5  
`Environment` (class in `rattlepy.environment`), 9  
`escapeHtmlEntities()` (in module `rattlepy.templating`), 5  
`exposes()` (`rattlepy.templating.Element` method), 5

**F**

`footer()` (in module `rattlepy.elements`), 3

**H**

`h1()` (in module `rattlepy.elements`), 3  
`h2()` (in module `rattlepy.elements`), 3  
`h3()` (in module `rattlepy.elements`), 3  
`h4()` (in module `rattlepy.elements`), 3  
`h5()` (in module `rattlepy.elements`), 3  
`h6()` (in module `rattlepy.elements`), 3  
`head()` (in module `rattlepy.elements`), 3  
`header()` (in module `rattlepy.elements`), 4

`hr()` (in module `rattlepy.elements`), 4  
`html()` (in module `rattlepy.elements`), 4

**I**

`img()` (in module `rattlepy.elements`), 4

**L**

`li()` (in module `rattlepy.elements`), 4  
`link()` (in module `rattlepy.elements`), 4

**M**

`main()` (in module `rattlepy.elements`), 4  
`meta()` (in module `rattlepy.elements`), 4

**N**

`node()` (in module `rattlepy.templating`), 6

**O**

`ol()` (in module `rattlepy.elements`), 4

**P**

`p()` (in module `rattlepy.elements`), 4  
`Placeholder` (class in `rattlepy.environment.placeholder`), 9

**R**

`rattlepy` (module), 1  
`rattlepy.elements` (module), 3  
`rattlepy.environment` (module), 9  
`rattlepy.environment.placeholder` (module), 9  
`rattlepy.templating` (module), 5  
`rattlepy.utils` (module), 7

**S**

`scaffold()` (in module `rattlepy.utils`), 7  
`script()` (in module `rattlepy.elements`), 4  
`SelfClosedElement` (class in `rattlepy.templating`), 6

`serialize()` (*rattlepy.templating.Element method*), 6  
`setTitle()` (*in module rattlepy.elements*), 4  
`span()` (*in module rattlepy.elements*), 4  
`style()` (*in module rattlepy.elements*), 4

## T

`text()` (*in module rattlepy.templating*), 6  
`title()` (*in module rattlepy.elements*), 4

## U

`ul()` (*in module rattlepy.elements*), 4