Attributes

Tag attributes look similar to html, however their values are just regular JavaScript.

```
a(href='google.com') Google
a(class='button', href='google.com') Google
```

```
<a href="google.com">Google</a><a href="google.com" class="button">Goo
```

All the normal JavaScript expressions work fine too:

```
- var authenticated = true
body(class=authenticated ? 'authed' : 'anon')
```

```
<body class="authed"></body>
```

If you have many attributes, you can also spread them across many lines:

```
input(
  type='checkbox'
  name='agreement'
  checked
)
```

```
<input type="checkbox" name="agreement" checked="checked"/>
```

Unescaped Attributes

By default, all attributes are escaped (replacing special characters with escape sequences) to prevent attacks such as cross site scripting. If you need to use special characters you can use `!=` instead of `=`.

```
div(escaped="<code>")
div(unescaped!="<code>")
```

```
<div escaped="&lt;code&gt;"></div>
<div unescaped="<code>"></div>
```

Danger

Unescaped buffered code can be dangerous. You must be sure to sanatize any user inputs to avoid Cross Site Scripting (http://en.wikipedia.org/wiki/Cross-site_scripting)

Boolean Attributes

Boolean attributes are mirrored by Jade, and accept bools, aka `true` or `false`. When no value is specified true is assumed.

```
input(type='checkbox', checked)
input(type='checkbox', checked=true)
input(type='checkbox', checked=false)
input(type='checkbox', checked=true.toString())
```

```
<input type="checkbox" checked="checked"/>
<input type="checkbox" checked="checked"/>
<input type="checkbox"/>
<input type="checkbox" checked="true"/>
```

If the doctype is `html` jade knows not to mirror the attribute and uses the terse style (understood by all browsers).

```
doctype html
input(type='checkbox', checked)
input(type='checkbox', checked=true)
input(type='checkbox', checked=false)
input(type='checkbox', checked=true && 'checked')
```

```
<!DOCTYPE html>
<input type="checkbox" checked>
<input type="checkbox" checked>
<input type="checkbox">
<input type="checkbox" checked="checked">
```

Class Attributes

The `class` attribute can be a string (like any normal attribute) but it can also be an array of class names, which is handy when generated from JavaScript.

```
- var classes = ['foo', 'bar', 'baz']
a(class=classes)
//- the class attribute may also be repeated to merge arrays
a.bing(class=classes class=['bing'])
```

```
<a class="foo bar baz"></a><a class="bing foo bar baz bing"></a>
```

Class Literal

Classes may be defined using a `.classname` syntax:

```
a.button
```

```
<a class="button"></a>
```

Since div's are such a common choice of tag, it is the default if you omit the tag name:

```
.content
```

```
<div class="content"></div>
```

ID Literal

IDs may be defined using a `#idname` syntax:

```
a#main-link
```

```
<a id="main-link"></a>
```

Since div's are such a common choice of tag, it is the default if you omit the tag name:

```
#content
```

```
<div id="content"></div>
```

&attributes

Pronounced "and attributes", the `&attributes` syntax can be used to explode an object into attributes of an element.

```
div#foo(data-bar="foo")&attributes({'data-foo': 'bar'})
```

```
<div id="foo" data-bar="foo" data-foo="bar"></div>
```

The object does not have to be an object literal. It can also just be a variable that has an object as its value (see also Mixin Attributes (/reference/mixins#attributes))

```
- var attributes = {'data-foo': 'bar'};
div#foo(data-bar="foo")&attributes(attributes)
```

```
<div id="foo" data-bar="foo" data-foo="bar"></div>
```

Danger

Attributes applied using `&attributes` are not automatically escaped. You must be sure to sanatize any user inputs to avoid Cross Site Scripting (http://en.wikipedia.org/wiki/Cross-site_scripting). This is done for you if you are passing in `attributes` from a mixin call.