Mixins

Mixins allow you to create reusable blocks of Pug.

```
//- Declaration
mixin list
  ul
    li foo
    li bar
    li baz
//- Use
+list
+list
```

```
<ul>
  <li>foo</li>
  <li>bar</li>
  <li>baz</li>
</ul>
<ul>
  <li>foo</li>
  <li>bar</li>
  <li>baz</li>
</ul>
```

Mixins are compiled to functions, and can take arguments:

```
mixin pet(name)
  li.pet= name
ul
  +pet('cat')
  +pet('dog')
  +pet('pig')
```

```
<ul>
  <li class="pet">cat</li>
  <li class="pet">dog</li>
  <li class="pet">pig</li>
```

```
</ul>
```

Mixin Blocks ¶

Mixins can also take a block of Pug to act as the content:

```
mixin article(title)
  .article
    .article-wrapper
      h1= title
      if block
        block
      else
        p No content provided

+article('Hello world')

+article('Hello world')
  p This is my
  p Amazing article
```

```
<div class="article">
  <div class="article-wrapper">
    <h1>Hello world</h1>
    <p>No content provided</p>
  </div>
</div>
<div class="article">
  <div class="article-wrapper">
    <h1>Hello world</h1>
    <p>This is my</p>
    <p>Amazing article</p>
  </div>
</div>
```

Mixin Attributes ¶

Mixins also get an implicit `attributes` argument, which is taken from the attributes passed to the mixin:

```
mixin link(href, name)
  //- attributes == {class: "btn"}
  a(class!=attributes.class href=href)= name

+link('/foo', 'foo')(class="btn")
```

```
<a class="btn" href="/foo">foo</a>
```

Note

The values in `attributes` by default are already escaped! You should use `!=` to avoid escaping them a second time. (See also unescaped attributes.)

You can also use mixins with `&attributes`:

```
mixin link(href, name)
  a(href=href)&attributes(attributes)= name

+link('/foo', 'foo')(class="btn")
```

```
<a class="btn" href="/foo">foo</a>
```

Note

The syntax `+link(class="btn")` is also valid and equivalent to `+link()(class="btn")`, since Pug tries to detect if parentheses' contents are attributes or arguments. Nevertheless, we encourage you to use the second syntax, as you pass explicitly no arguments and you ensure the first parenthesis is the arguments list.

Default Argument's Values ¶

You can also set default values for you `arguments`. Same as setting *default function parameters* in ES6.

```
//- Declaration
mixin article(title='Default Title')
  .article
    .article-wrapper
      h1= title
```

```
//- Use
+article()

+article('Hello world')
```

```
<div class="article">
  <div class="article-wrapper">
    <h1>Default Title</h1>
  </div>
</div>
<div class="article">
  <div class="article-wrapper">
    <h1>Hello world</h1>
  </div>
</div>
```

Rest Arguments ¶

You can write mixins that take an unknown number of arguments using the "rest arguments" syntax.

```
mixin list(id, ...items)
  ul(id=id)
    each item in items
      li= item

+list('my-list', 1, 2, 3, 4)
```

```
<ul id="my-list">
  <li>1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
</ul>
```

Get an update when we release new features

Name (optional):

Email:

Sign Up

We respect your email privacy