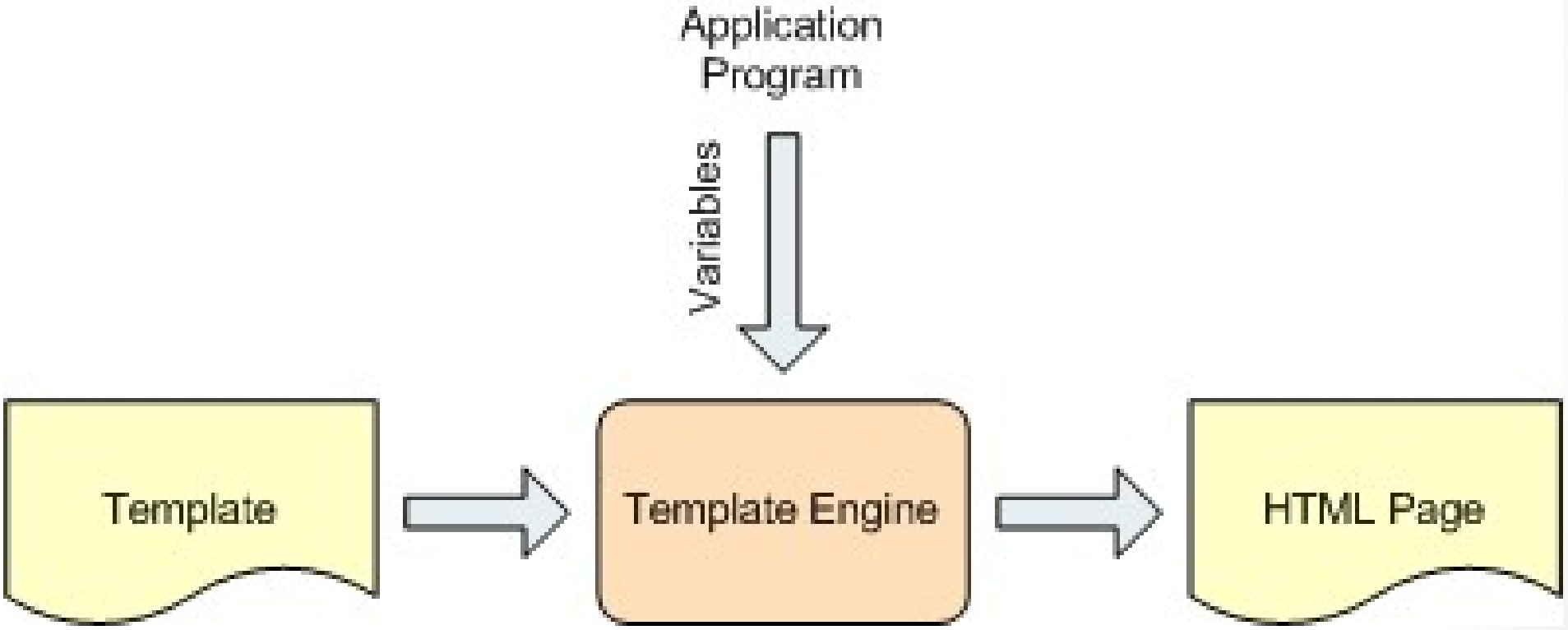
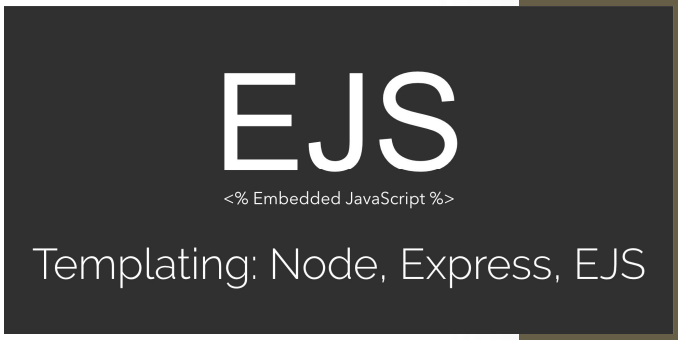
pug

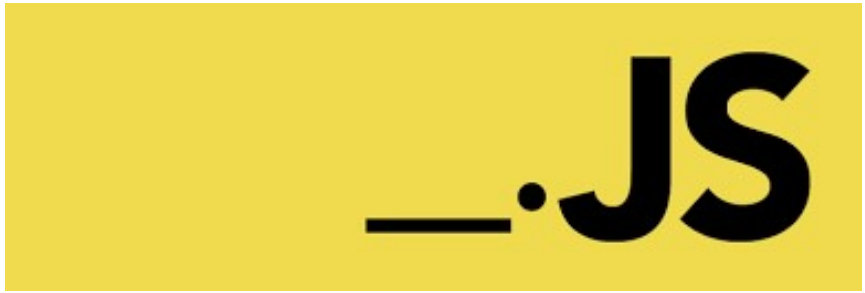
Fernando Saez

Que es un motor de plantillas



Opciones





PUG anteriormente Jade y EJS son los motores mas populares.

Pug

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript').
      if (foo) {
        bar(1 + 5)
      }
  body
    h1 Jade - node template engine
    #container.col
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
      p.
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
```

EJS

```
<!DOCTYPE html>
<html lang="en">
<body class="container">

    <h1>Some Title</h1>

    <% if (names.length) { %>
      <ul>
        <% names.forEach(function(name){ %>
          <li><%= name %></li>
        <% }) %>
      </ul>
    <% } %>

    <p>foo bar</p>

</body>
</html>
```

Pug

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript').
      if (foo) {
        bar(1 + 5)
      }
  body
    h1 Jade - node template engine
    #container.col
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
      p.
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
```

Template

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
    <script type="text/javascript">
      if (foo) {
        bar(1 + 5)
      }
    </script>
  </head>
  <body>
    <h1>Jade - node template engine</h1>
    <div id="container" class="col">
      <p>You are amazing</p>
      <p>
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
      </p>
    </div>
  </body>
</html>
```

Salida

Que es Pug (Jade)

PUG es un motor de templates de muy alta-performance.

Trabaja como un intermediario entre la interfaz de entrada y la respuesta producida.

PUG ayuda a renderizar el Código HTML basandose en las entradas del usuario o los datos de referencia.

Es un pre-procesador que ayuda a facilitar las tareas repetitivas proviendo características que no están disponibles en HTML.

Como usar pug

- >npm install pug

```
//- template.pug
p Código fuente en pug de #{name}
```

```
//- index.js
const pug = require('pug’);
//compila el archivo
const compiledFunction = pug.compileFile('template.pug’);
// Renderiza los datos pasados
console.log(compiledFunction({name: ‘Juan Perez'}));
//imprime  "<p> Código fuente en pug de Juan Perez</p>“

console.log(compiledFunction({name: ‘Ana Tussi'}));
//imprime  "<p> Código fuente en pug de Ana Tussi</p>“
```

Renderizar pug

```
//- template.pug
p Código fuente en pug de #{name}
```

```
//index.js
const pug = require('pug');

// Compile template.pug, and render a set of data
console.log(pug.renderFile('template.pug', {
  name: 'Juan Perez'
}));
// "<p> Código fuente en pug de Juan Perez</p>"
```

Elementos o Tags

```
ul
  li Item A
  li Item B
  li Item C
```

```
<ul>
  <li>Item A</li>
  <li>Item B</li>
  <li>Item C</li>
</ul>
```

```
a: img
```

```
<a><img /></a>
```

```
foo/
foo(bar='baz')/
```

```
<foo />
<foo bar="baz" />
```

Plain Text

1

```
p This is plain old <em>text</em> content.
```

```
<p>This is plain old <em>text</em> content.</p>
```

2

```
<html>

body
  p indentar el body aquí no hace diferencia.
  p HTML no es sensible a espacios en blanco.

</html>
```

```
<html>

<body>
  <p> indentar el body aquí no hace diferencia.</p>
  <p>HTML no es sensible a espacios en blanco.</p>
</body>

</html>
```

Plain Text (2)

3

```
p
  | El pipe siempre va en el comienzo de
la línea,
  | no cuenta la indentación.
```

```
<p> El pipe siempre va en el comienzo de la
línea,
 no cuenta la indentación.</p>
```

4

```
script.
  if (usingPug)
    console.log('you are awesome')
  else
    console.log('use pug')
```

```
<script>
  if (usingPug)
    console.log('you are awesome')
  else
    console.log('use pug')
</script>
```

Control de espacios en blanco

1

```
| Ponemos em
strong fa
| sis sobre la si
strong la
| ba.
```

```
Ponemos em<strong>fa</strong>sis sobre la
si<strong>la</strong>ba.
```

Agregar espacios en blanco

2

```
| No
|
button#self-destruct toques
|
| este botón
```

```
No

<button id="self-destruct">
toques</button>
Este botón
```

Agregar Código JS sin buffer

```
- for (var x = 0; x < 3; x++)
  li item
```

```
<li>item</li>
<li>item</li>
<li>item</li>
```

```
- var list = ["Uno", "Dos", "Tres",
          "Cuatro", "Cinco", "Seis"]
- for (var y = 0; y < list.length ; y++)
  li
    = list[y]
```

```
<li>Uno</li>
<li>Dos</li>
<li>Tres</li>
<li>Cuatro</li>
<li>Cinco</li>
<li>Seis</li>
```

Agregar Código JS con buffer

```
p
  = 'This code is <escaped>!'
```

```
<p>This code is &lt;escaped&gt;!</p>
```

```
p= 'This code is' + ' <escaped>!'
p(style="background: blue")= 'A message with a ' + 'blue' + ' background'
```

```
<p>This code is &lt;escaped&gt;!</p>
<p style="background: blue">A message with a blue background</p>
```

Código con buffer sin escape

```
p
  != 'This code is <strong>not</strong> escaped!'
```

```
<p>This code is <strong>not</strong> escaped!</p>
```

Interpolación

```
- var titulo = “El perro es el mejor amigo del hombre";
- var autor = “Cris";
- var genial = "<span>escape!</span>";

h1= titulo
p Escrito con amor por #{autor}
p Este texto será seguro: #{genial}

<!– Salida -- >
<h1>El perro es el major amigo del hombre</h1>
<p>Escrito con amor por Cris</p>
<p>Este texto sera Seguro : &lt;span&gt;escape!&lt;/span&gt;</p>
```

Interpolación (2)

```
- var msg = “No es mi voz";
p Esto #{msg.toUpperCase()}
```

```
<p>Esto NO ES MI VOZ</p>
```

```
- var atencion = "<em>Hay alguien ahí.</em>";
.quote
  p Joel: !{atencion}
```

```
<div class="quote">
  <p>Joel: <em> Hay alguien ahí.</em></p>
</div>
```

Atributos

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

```
<!—Salida→
<a href="//google.com">Google</a>
<a class="button" href="//google.com">Google</a>
<a class="button" href="//google.com">Google</a>
```

Atributos (2)

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

<!—Salida→
<body class="authed"></body>
```

```
input(
  type='checkbox'
  name='agreement'
  checked
)
<!—Salida→
<input type="checkbox" name="agreement" checked="checked"
/>
```

Atributos (3)

```
- var url = 'pug-test.html';
a(href='/' + url) Link
|
|
- url = 'https://example.com/'
a(href=url) Another link
<!—Salida→
<a href="/pug-test.html">Link</a>
<a href="https://example.com/">Another link</a>
```

```
- var btnType = 'info'
- var btnSize = 'lg'
button(type='button' class=`btn btn-${btnType} btn-${btnSize}`)
<!—Salida→
<button type="button” class="btn btn-info btn-lg"></button>
```

Atributos (4)

```
a(style={color: 'red', background: 'green'})
<!—Salida→
<a style="color:red;background:green;"></a>
```

```
- var classes = ['foo', 'bar', 'baz']
a(class=classes)
|
|
//- the class attribute may also be repeated to merge arrays
a.bang(class=classes class=['bing’])
<!—Salida→
<a class="foo bar baz"></a>
<a class="bang foo bar baz bing"></a>
```

Atributos (5) (Classname sintax)

a.button	
.content	
a#main-link	
#content	

```
- var attributes = {};
- attributes.class = 'baz';
- attributes.enabled = false
div#foo(data-bar="foo")&attributes(attributes)

<!—Salida→
<div class="baz" id="foo" data-bar="foo" enabled=false></div>
```

Include

```
//- index.pug
doctype html
html
  include includes/head.pug
  body
    h1 Mi sitio
    p Bienvenidos a mi sitio.
    include includes/foot.pug
```

```
//- includes/head.pug
head
  title Mi pagina pug
  script(src='/scripts/jquery.js')
  script(src='/scripts/app.js')
```

```
//- includes/foot.pug
footer#footer
  p Copyright (c) foobar
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Mi pagina pug</title>
  <script
src="/scripts/jquery.js"></script>
  <script
src="/scripts/app.js"></script>
</head>
<body>
  <h1>Mi sitio</h1>
  <p>Bienvenidos a mi sitio.</p>
  <footer id="footer">
    <p>Copyright (c) foobar</p>
  </footer>
</body>
</html>
```

Condicionales: Case

```
- var friends = 10
case friends
  when 0
    p you have no friends
  when 1
    p you have a friend
  default
    p you have #{friends} friends

<!—Salida→
<p>you have 10 friends</p>
```

```
- var friends = 1
case friends
  when 0: p you have no friends
  when 1: p you have a friend
  default: p you have #{friends}
friends

<!—Salida→
<p>you have a friend</p>
```

Condicional if else

```
- var user = {description: 'foo bar baz'}
- var authorised = false
#user
  if user.description
    h2.green Description
    p.description= user.description
  else if authorised
    h2.blue Description
    p.description.
      User has no description,
      why not add one...
  else
    h2.red Description
    p.description User has no description
```

```
<div id="user">
  <h2
class="green">Description</h2>
  <p   class="description">foo   bar
baz</p>
</div>
```

```
if !user.isAnonymous
  p You're logged in as #{user.name}
```

Equivalente

```
unless user.isAnonymous
  p You're logged in as #{user.name}
```

Iteraciones each

```
ul
  each val in [1, 2, 3, 4, 5]
    li= val
```

```
<ul>
  <li>1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
  <li>5</li>
</ul>
```

```
ul
  each val, index in ['zero', 'one',
  'two']
    li= index + ': ' + val
```

```
<ul>
  <li>0: zero</li>
  <li>1: one</li>
  <li>2: two</li>
</ul>
```

```
- var values = [];
ul
  each val in values
    li= val
  else
    li There are no values
```

```
<ul>
  <li>There are no values</li>
</ul>
```

Condicional – Iteraciones while

```
- var n = 0;
ul
  while n < 4
    li= n++
```

```
<ul>
  <li>0</li>
  <li>1</li>
  <li>2</li>
  <li>3</li>
</ul>
```

Mixins

```
//- 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>
```

```
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>
```

Mixins (2)

```
mixin article(title)
  .article
    .article-wrapper
      h1= title
      if block
        block
      else
        p Bloque no provisto

+article(`Hola Mundo')

+article(`Hola Mundo')
  p Esto es un bloque pasado
  p artículo
```

```
<div class="article">
  <div class="article-wrapper">
    <h1>Hello world</h1>
    <p>Bloque no provisto</p>
  </div>
</div>
<div class="article">
  <div class="article-wrapper">
    <h1>Hola Mundo</h1>
    <p>Esto es un bloque pasado</p>
    <p>artículo</p>
  </div>
</div>
```

Mixins (3)

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

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

<!—Salida→
<a class="btn" href="/foo">foo</a>
```

```
//- 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>
```