PUG

PUG

- Pug is a templating engine that uses template strings to create HTML.
- Has language aspects that allow it to modify the output of engine on the fly.
- You can find the documentation at https://pugjs.org/api/getting-started.html

PUG - Tags

- Text at the start of a line is typically a tag. Don’t use angle braces. Don’t need to close.
- Indented (2 spaces) tags are nested.
- Read more [Reference]

```
div Content

ol
  li Item 1
  li Item 2
  li Item 3
```

These are nested in the ol.

PUG – Plain text

- Put it on the same line as the tag. Can be html.
- Create a block using a . after the tag.
- Read more Reference

```
p Some text in <bold>my</bold> paragraph

p.
  This is
  Some text
  in a
  paragraph.
```

These are all inside the paragraph as its contents.

PUG – Plain text - Pipe

- If we start a line with a pipe | then it is combined with the previous. This is useful to mix text with inline tags.
- Also used for space control.

```
p.
  | This is
  | more text
  | in a
  | paragraph.
```

These are all inside the paragraph as its contents.

PUG – Plain text block

- If we add a . on the end of a tag the rest will just be rendered as plain text.
- Need to indent by one level.

```
script.
     const y = 10;
     function best (x) {
          if (x>y) then
          {
               x++;
               console.log(”keep going”)
          }
     }
```

This is all rendered as is and is convenient for adding in java script to be executed on the client side.

- Pug is sensitive to indentation and spaces.

PUG – Plain text interpolation

- Anything inside of #{} will be evaluated as a java script expression and escaped for safety.
- Read More Reference

```
- var x = 20
p We are going around #{x+2} times.
```

PUG - Code

- We can use inline JavaScript in our templates
- Unbuffered code starts with –. It does not directly add anything to the output of the renderer.
- Buffered code starts with = and will be HTML escaped. Whatever follows the = will be evaluated as a javascript expression.
- Unescaped buffered code starts with !=. It will not be escaped and is not safe for input.

PUG - Code

- Read more Reference

```
- for(var i=0; i<5; i++)
  li item
```

Five copies of <li> item </li>

```
- var list = ['one', 'two', 'three']
each item in list
  li= item
```

= triggers buffered evaluation of JS expression item which gives
<li>one</li>
<li>two</li>
<li>three</li>

PUG - Comments

- Comments start with // or //-
- // Comments are buffered and result in an HTML comment.
- If you want a multiline comment, leave the rest of the line blank
- Read more Reference

```
//-
    This is a multiline comment
    That is not part of the HTML output
//  This is a single line comment and is rendered
```

PUG - Attributes

- Attributes of elements are kept in ()
- We can use variables with concatenation or string interpolation with back-ticked strings.

```
div(id=”happy” onClink=“method()”) Content

- var btnType = 'info'
- var btnSize = 'lg’

button(type='button’
   class='btn btn-' + btnType + ' btn-' + btnSize
) Button 1

button(type='button’
   class=`btn btn-${btnType} btn-${btnSize}`
) Button 2
```

PUG - Attributes

- Style attributes can be strings or an object
- Class attributes can be strings or a list
- Can use a selector style with . or # defaulting to div
- Read more Reference

```
p(style = {color:"green", background:"blue"}) content

- var classes=['long', 'extra', 'foo']
div(class=classes) more text

a.button
a#my-link

div#contentid
#contentid
```

These two are the same. Div is so common that it is taken as default.

PUG - Conditionals

- Don’t need () on condition
- Don’t use - with if/else
- Read more Reference

```
- 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
```

Gives <div id=”user”>

One of three yellow chunks will be rendered depending on the if

Render after . as plain text.

PUG - Conditionals

```
- 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
```

We get the first with an inline evaluation of user.description.

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

PUG - Loops

- Special constructs each/while don't use – and are unbuffered
- Read more Reference

```
- var my_list = [1, 2, 4]
each value in my_list
  li= value

each value,index in my_list
  div item #{value*index}

- var part = 0
while part<10
  - part += 1
  p paragraph count is #{part}
```

Gives

```
  <li>1</li>
  <li>2</li>
  <li>4</li>
```

Gives

```
  <div>item 0</div>
  <div>item 2</div>
  <div>item 8</div>
```

PUG - Loops

- Can add an else clause to each for empty loops

```
- var my_list = []
each value in my_list
  li= value
else
  p No values in list
```

PUG - Case

- No – needed for case/when/default
- Fall through only on empty when.
- Read more Reference

```
- var value = 10
case value%3
  when 0
    p Divisible by 3
  when 1
  when 2
    p Not divisible by 3
  default
    p Non integer remainder for #{value}
```

Fall through on when 1.

PUG - Doctype

- There are others, but for modern HTML 5 we will use html

```
doctype html
```

PUG - Include

- Includes the contents of some other file.
- If no extension, it is assumed to be pug
- If some other kind of file, the text is just included directly without being rendered.

PUG - Inheritance

- In a parent template, block name defines a part of the template that can be replaced by the child.
- The child template extends the parent
- Instead of replacement, we can add to the parents definition
  - *block prepend name*
  - *block append name*
- Read more Reference

PUG - Mixin

- This effectively allows us to create a pug function that can be called to generate some HTML
- Read More Reference

```
mixin pet(name)
  li.pet= name

ul
  +pet('cat')
  +pet('dog')
  +pet('pig')
```

Nothing rendered yet. It will create a listitem with class pet. The contents of the item will be interpolated.

+ is the mix-in “call”.

```
<ul>
  <li class=“pet”>cat</li>
  <li class=“pet”>dog</li>
  <li class=“pet”>pig</li>
<\ul>
```

Routes – index.js Revisited

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
      res.render('index', { title: 'Express' });
});

module.exports = router;
```

Variable title is set to ‘Express’

View – layout.pug Revisited

```
doctype html
  html
    head
      title= title
      link(rel='stylesheet', href='/stylesheets/style.css’)
    body
      block content
```

Evaluate the expression title

Parent. Could have defined default content to place here

View – index.pug Revisited

```
extends layout

block content
  h1= title
  p Welcome to #{title}
```

Child

Replace block content of parent

Uses interpolation for content

app.js Revisited

Set local variables that are visible to the Pug render engine

```
// error handler
app.use(function(err, req, res, next) {
   // set locals, only providing error in
development
   res.locals.message = err.message;
   res.locals.error = req.app.get('env') ===
'development' ? err : {};

   // render the error page
   res.status(err.status || 500);
   res.render('error');
});

module.exports = app;
```

Will use the error view

View – error.pug

```
extends layout

block content
  h1= message
  h2= error.status
  pre #{error.stack}
```

Child

error and message are locals that can be evaluated and used to set the content.