DUST.JS AT PAYPAL

WHAT IS DUST.JS?

- Templating system for Node.js and the browser
- Extended Mustache-style syntax
- Compile-to-JavaScript for speed and portability
- Streaming rendering

LINKEDIN FORK OF DUST.JS

http://linkedin.github.io/dustjs/

```
$ npm install dustjs-linkedin
```

LET'S SEE WHAT DUST CAN DO

01_duster.js — a command-line Dust.js compiler

```
'use strict';

if (process.argv.length === 2)
  return console.log('Usage: node 01_duster.js <template.dust> [data.jso

var dust = require('dustjs-linkedin'),
    fs = require('fs'),
    path = require('path'),
    // load Dust template file and optional data JSON file
    tmpl = fs.readFileSync(process.argv[2], 'utf8'),
    data = process.argv.length > 3 ?
           require(path.resolve(process.argv[3])) : {},
    // compile template to JavaScript string with ID 'cmdline'
    compiled = dust.compile(tmpl, 'cmdline');

// register 'cmdline' template with the Dust engine
dust.loadSource(compiled);

console.log('Compiled JavaScript:\n\n', compiled);
console.log('\n-------------------------------------------------------')
console.log('Rendered template:\n');

// render 'cmdline' template with the context data
dust.render('cmdline', data, function (err, out) {
  if (err) throw err;
  console.log(out);
});
```

FIRST DUST TEMPLATE

02_example1.dust

```
<h1>{title}</h1>
<ul>
{#names}
  <li><a href="{url}">{name}</a></li>{~n}
{/names}
</ul>
```

03_example1.json

```
{
  "title": "Node.js Core Dependencies",
  "names": [
    { "name": "libuv", "url": "https://github.com/joyent/libuv" },
    { "name": "V8", "url": "http://code.google.com/p/v8/" },
    { "name": "OpenSSL", "url": "http://www.openssl.org/" },
    { "name": "zlib", "url": "http://www.zlib.net/" },
    { "name": "c-ares", "url": "http://c-ares.haxx.se/" },
    { "name": "http_parser", "url": "https://github.com/joyent/http-pars
  ]
}
```

```
$ node 01_duster.js 02_example1.dust 03_example1.json
```

```
<h1>Node.js Core Dependencies</h1><ul><li><a href="https://github.com/jo
<li><a href="http://code.google.com/p/v8/">V8</a></li>
<li><a href="http://www.openssl.org/">OpenSSL</a></li>
<li><a href="http://www.zlib.net/">zlib</a></li>
<li><a href="http://c-ares.haxx.se/">c-ares</a></li>
<li><a href="https://github.com/joyent/http-parser">http_parser</a></li>
</ul>
```

DUST BASICS

- "keys": `{title}` — look up variables in the context data
- "sections": `{#names} ... {/names}` — can be used for iteration, or scope capturing
- "partials": `{> "header" /}` — include one template in another
- "comments": `{! comments !}` — not rendered
- "helpers": `{@idx}` — for more complex logic and custom plugins

KEYS

- `{name}` looks up the property `"name"` in the current context
- Context is the root passed in to the renderer but is changed when entering "sections"
- Keyed property has no value? No output
- Output values are escaped by default to prevent XSS

02_example1.dust

```
<h1>{title}</h1>
<ul>
{#names}
  <li><a href="{url}">{name}</a></li>{~n}
{/names}
</ul>
```

KEYS AND FILTERS

- Filters can be applied:
  - `{name|s}` suppresses auto-escaping
  - `{name|h}` force HTML escaping
  - `{name|j}` force JavaScript escaping
  - `{name|u}` encodes with JS native encodeURI
  - `{name|uc}` encodes with JS native encodeURIComponent
  - `{name|js}` stringify JSON literal
  - `{name|jp}` parse JSON string to object
  - Filters can be chained: `{name|s|h}`
- Dust agressively removes unnecessary white-space but special characters can be inserted:
  `{~n}` newline, `{~r}` CR, `{~lb}` left bracket, `{~rb}` right bracket, `{~s}` - space

SECTIONS

04_friends.dust

```
<ul>{~n}
{#friends}
  <li>(#{$idx} of {$len}) {name}, {country}</li>{~n}
{:else}
  <p>You have no friends!</p>
{/friends}
</ul>
```

- Can be used to iterate over arrays
- Inside the *section*, the context is set to the current item of the array
  - e.g. `{#friends} {name} {/friends}` will display the `"name"` property for each element of the "friends" array
- If a key inside a section doesn't reference data on that item, it will look in the parent context
- `{:else}` block will be printed if there are no elements
- `{$idx}` is the current index (zero-based), `{$len}` is the number of total elements

```
$ node 06_duster2.js 04_friends.dust 05_friends.json
```

PARTIALS

- Templates are registered with Dust by name and can be referenced within other templates to include their contents
- References to partials can be *parameterized* with contextual data

PARTIALS

06_duster2.js — modified to process multiple dust templates

```
'use strict';

if (process.argv.length < 4)
  return console.log(
      'Usage: node 06_duster2.js <mainTemplate.dust>[, <template2.dust>[

var dust = require('dustjs-linkedin'),
    fs = require('fs'),
    path = require('path'),
    // load JSON data from last arg
    data = require(path.resolve(process.argv.pop())),
    // Dust template names are the other args
    tmplNames = process.argv.slice(2),
    mainId; // main template ID, the first one in the list

mainId = tmplNames.map(function (t) {
  // register each template with filename (minus extension) as ID
  var id = t.replace(/\..+$/, '');
  var contents = fs.readFileSync(t, 'utf8');
  var compiled = dust.compile(contents, id);
  dust.loadSource(compiled);
  return id;
})[0]

dust.render(mainId, data, function (err, out) {
  if (err) throw err;
  console.log(out);
});
```

PARTIALS

07_example2.dust

```
<p>
  I am {name} and I live in
  {~s}
  {> "08_display_address" address=homeAddress /}
</p>
```

08_display_address.dust

```
<address>{address.city}, {address.state}</address>
```

09_example2.json

```
{
  "name": "Phineas Gage",
  "homeAddress": {
    "city": "Cavendish",
    "state": "Vermont"
  }
}
```

```
$ node 06_duster2.js 07_example2.dust 08_display_address.dust 09_example
```

```
<p>I am Phineas Gage and I live in <address>Cavendish, Vermont</address>
```

PARTIALS

Block substitution in partials

10_header.dust

```
<head>
  {! "Named block" with a default value !}
  <title>{+pageTitle}{defaultTitle}{/pageTitle}</title>
</head>
```

11_about.dust

```
{! Include partial !}
{> "10_header" /}

{! Populate named block !}
{<pageTitle}About Us{/pageTitle}

<body>Here's some information about us</body>
```

12_data.json

```
{ "defaultTitle": "PayPal" }
```

Render the template and partial:

```
$ node 06_duster2.js 11_about.dust 10_header.dust 12_data.json
```

```
<head><title>About Us</title></head><body>Here's some information about
```

LOGIC

Sections can be used to test **existential** logic (not just lists!)

14_logic.dust

```
My name is {name} and I:
<ul>
       {! Sections are used here to check if the variable is "true", not
       {! We are also using "." to reference properties of a parent obje
  <li> {#possessions.car} own a {.} {:else} don't own a {/possessions.ca
  <li> {#possessions.motorbike} own a {.} {:else} don't own a {/possessi
  <li> {#possessions.bicycle} own a {.} {:else} don't own a {/possession
</ul>
```

15_logic.json

```
{
  "name": "Roger",
  "possessions": {
    "car": "red",
    "bicycle": "silver"
  }
}
```

```
$ node 06_duster2.js 14_logic.dust 15_logic.json
```

```
My name is Roger and I:
<ul>
  <li>own a red car </li>
  <li>don't own a motorbike </li>
  <li>own a silver bicycle </li>
</ul>
```

LOGIC

- Dust "true" and "false" are different to "truthy" and "falsey" in JavaScript:
  - **"true"**: 0, "0", "null", "undefined", "false", {} (empty object)
  - **"false"**: "", " ", `false`, null, undefined, [] (empty array)

LOGIC

- `{?name} body {/name}` — will check if "name" evaluates to "true", not just exists
- `{^name} body {/name}` — will check if "name" evaluates to "false"

HELPERS

Standard helpers are shipped in a separate package:

```
$ npm install dustjs-helpers
```

Load them into the Dust runtime:

```
var dust = require('dustjs-linkedin');

require('dustjs-helpers');
```

HELPERS

Use `{@select}` like a switch / case statement

```
{@select key="{foo}"}
   {@eq value="bar"}foobar{/eq}
   {@eq value="baz"}foobaz{/eq}
   {@default} - default Text{/default}
{/select}

{@select key=foo}
   {@gte value=5}foobar{/gte}
{/select}
```

@eq, @ne, @lt, @lte, @gt, @gte, are available and can be used outside @select:

```
<select name="courses">
  {#options}
    <option value="{value}" {@eq key=value value=courseName} selected="t
  {/options}
</select>
```

HELPERS

{@math}

```
{@math key="16" method="add" operand="4"/}  - Result will be 20
{@math key="16.5" method="floor"/} - Result will be 16
{@math key="16.5" method="ceil"/} - Result will be 17
{@math key="-8" method="abs"/} - Result will be 8
{@math key="{$idx}" method="mod" operand="2"/} - Return 0 or 1 according
```

{@if}

```
{@if cond="{x} < {y} && {b} == {c} && '{e}'.length || '{f}'.length"}
<div> x is less than y and b == c and either e or f exists in the output
{/if}

{@if cond="({x} < {y}) || ({x} < 3)"} <div> x<y and x<3 {/if}

{@if cond="{x} < {y} && {b} == {c} && '{e}'.length || '{f}'.length "}
 <div> x is less than y and b == c and either e or f exists in the outpu
{:else}
 <div> x is >= y </div>
{/if}
```

{@sep} to prevent dangling-commas:

```
{#friends}
 {name}{@sep},{/sep}
{/friends}
```

DUST.JS AND ADARO

- Dust view renderer for Express
- Layout support
- Helpers support
- Cache control
- Custom pipeline

ADARO LAYOUTS

```
// Use alternate layout
dust.render('index', { layout: 'myLayout' }, ...);
```

```
// Disable layout altogether
dust.render('index', { layout: false }, ...);
```

HOW ADARO LAYOUTS WORK IN KRAKEN.JS

Layout uses dynamic `{_main}` property to look up the current content "partial"

```
<html lang="en">
  <body>
    <div id="wrapper">
      {>"{_main}"/}
    </div>
  </body>
</html>
```

Content page (view) is just a partial with no knowledge of the layout

```
<h1>Hello World!</h1>
```

USING MAKARA

i18n on top of Adaro

MAKARA: DUST LOCALIZATION HELPERS

Pre-processed tags replaced with localized values prior to full Dust rendering

```
{@pre type="content" key="index.title" /}
```

Properties files

```
index.title=PayPal Merchant
index.greeting=Welcome {userName}
```

MAKARA PROPERTIES FILES

US/en/index.properties - localized properties

```
index.title=PayPal Merchant
index.greeting=Welcome {userName}

# A list
index.ccList[0]=Visa
index.ccList[1]=Mastercard
index.ccList[2]=Discover

# A map
index.states[AL]=Alabama
index.states[AK]=Alaska
index.states[AZ]=Arizona
index.states[CA]=California
```

AU/en/index.properties - localized properties

```
index.title=PayPal Australia Merchant
index.greeting=Welcome {userName}

# A list
index.ccList[0]=Visa
index.ccList[1]=Mastercard

# A map
index.states[NSW]=New South Wales
index.states[VIC]=Victoria
index.states[SA]=South Australia
index.states[QLD]=Queensland
```

16_makara.js — using Makara directly

```
var i18n = require('makara'),
    provider = i18n.create({
      contentPath: __dirname,
      fallback: 'en_US'
    });

function printIndex (locale, callback) {
  provider.getBundle('index', locale, function (err, bundle) {
    if (err) return callback(err);

    'title greeting ccList states'.split(' ').forEach(function (key) {
      console.log(locale, key, '=', bundle.get('index.' + key));
    })

    callback();
  });
}

var locales = 'en_US en_AU'.split(' '); // what happens if we add more?
var i = 0;

(function next () {
  printIndex(locales[i], function (err) {
    if (err) throw err;
    if (++i < locales.length)
      next(); // loop again with next locale
  });
}()); // immediately invoked function
```

```
$ npm install makara
$ node 16_makara.js
```

USING MAKARA WITH EXPRESS

17_makara_express.js

```
var express = require('express'),
    i18n = require('makara'),
    dustjs = require('adaro');

var app = express();

app.engine('dust', dustjs.dust());
app.set('views', 'templates/');
app.set('view engine', 'dust');

app.get('/', function (req, res) {
  // render "index" with a "userName" property
  res.render('index', { userName: 'dshaw' });
});

// decorate express app with Makara
i18n.create(app, {
  contentPath: __dirname,
  fallback: 'en_US'
});

app.listen(3000, function () {
  console.log('Listening on http://localhost:3000/');
});
```

```
$ npm install express adaro
$ node .
```

FINDATAG

- Used as a pre-processor in Makara to expand localized strings
- Streaming string tokenizer for recognizing Dust tags: `{@tag ... /}`

USING FINDATAG DIRECTLY

18_findatag.js

```
var finder = require('findatag'),
    fs = require('fs');

/// Entity handler

var finderStream = finder.createParseStream(entityHandler);
fs.createReadStream('tags_in_a_haystack.dust').pipe(finderStream).pipe(p
```

USING FINDATAG DIRECTLY

18_findatag.js

```
/// Entity handler
var entityHandler = {
  tagHandlers: {
     'needle': {
        exec: function (def, callback) {
          // callback sends the data we want to send out of the stream i
          callback(null, '\nneedle: ' + JSON.stringify(def, null, 2) + '
        }
      },
     'squid': {
        exec: function (def, callback) {
          callback(null, '\nsquid: ' + JSON.stringify(def, null, 2) + '\
        }
      }
  },

  get tags () { // list of tags we want to know about
    return Object.keys(this.tagHandlers);
  },

  onTag: function (def, callback) { // when a tag is found in a stream
    this.tagHandlers[def.name].exec(def, callback);
  },

  onText: function (chunk, callback) { // when everything else is found
    callback(null, ''); // return an empty string, no output here
  }
}
```

USING FINDATAG DIRECTLY

```
$ npm install findatag
$ node 18_findatag.js
```

SUMMARY

- Dust.js is a flexible and fast templating engine
  - Docs: http://linkedin.github.io/dustjs/
- Adaro connects Dust views and layout functionality to Express
- Makara is a Dust helper for i18n
- Makara uses findatag to parse templates