meltingice / **psd.js**

A Photoshop PSD file parser for NodeJS and browsers

184 commits **10** branches **0** packages **4** releases **12** contributors MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

fabulousduck Merge pull request #164 from sngjuk/patch-2 … ✓ Latest commit 333dd14 on 3 Jul 2019

dist	fix it to read multiple object effect	12 months ago
docs	Move docs to top of psd.coffee for better layout	5 years ago
examples	Update layer_comps.js	4 years ago
lib	key comparison syntax changed	12 months ago
shims	Fix toPng so that it's no longer asynchronous #19	5 years ago
test	fix broken test	3 years ago
.gitignore	Add greyscale image exporting	5 years ago
.npmignore	Ignore docs in npm publish	5 years ago
.travis.yml	up travis build versions to prevent old nodejs bugs	12 months ago
Cakefile	Forgot to git push in docs deploy	5 years ago
LICENSE.txt	Create LICENSE.txt	6 years ago
README.md	[ci skip] Get more Open Source Helpers	3 years ago
bower.json	Ignore unneeded folders for bower	5 years ago
index.js	First working example with browserify	6 years ago
package.json	Move acorn to devDependencies	3 years ago

README.md

PSD.js

Build Status Help Contribute to Open Source

A general purpose PSD parser written in Coffeescript. Based off of PSD.rb. It allows you to work with a Photoshop document in a manageable tree structure and find out important data such as:

- Document structure
- Document size
- Layer/folder size + positioning

- Layer/folder names
- Layer/folder visibility and opacity
- Font data (via psd-enginedata)
  - Text area contents
  - Font names, sizes, and colors
- Color mode and bit-depth
- Vector mask data
- Flattened image data
- Layer comps

Runs in both NodeJS and the browser (using browserify). There are still some pieces missing that are present in PSD.rb, such as layer comp filtering, a built-in renderer, and many layer info blocks. The eventual goal is full feature parity with PSD.rb.

Installation

PSD.js has no native dependencies. Simply add `psd` to your package.json or run `npm install psd`.

Documentation

Note: work in progress

Annotated source code documentation is available here. PROTIP: if you're wondering how to access various metadata from a layer, you'll want to see this file.

Usage

PSD.js works almost exactly the same in the browser and NodeJS.

NodeJS Example

```
var PSD = require('psd');
var psd = PSD.fromFile("path/to/file.psd");
psd.parse();

console.log(psd.tree().export());
console.log(psd.tree().childrenAtPath('A/B/C')[0].export());

// You can also use promises syntax for opening and parsing
PSD.open("path/to/file.psd").then(function (psd) {
  return psd.image.saveAsPng('./output.png');
}).then(function () {
  console.log("Finished!");
});
```

Browser Example

```
var PSD = require('psd');

// Load from URL
PSD.fromURL("/path/to/file.psd").then(function(psd) {
  document.getElementById('ImageContainer').appendChild(psd.image.toPng());
});

// Load from event, e.g. drag & drop
function onDrop(evt) {
  PSD.fromEvent(evt).then(function (psd) {
    console.log(psd.tree().export());
  });
}
```

Traversing the Document

To access the document as a tree structure, use `psd.tree()` to get the root node. From there, work with the tree using any of these methods:

- `root()` : get the root node from anywhere in the tree
- `isRoot()` : is this the root node?
- `children()` : get all immediate children of the node
- `hasChildren()` : does this node have any children?
- `childless()` : opposite of `hasChildren()`
- `ancestors()` : get all ancestors in the path of this node (excluding the root)
- `siblings()` : get all sibling tree nodes including the current one (e.g. all layers in a folder)
- `nextSibling()` : gets the sibling immediately following the current node
- `prevSibling()` : gets the sibling immediately before the current node
- `hasSiblings()` : does this node have any siblings?
- `onlyChild()` : opposite of `hasSiblings()`
- `descendants()` : get all descendant nodes not including the current one
- `subtree()` : same as descendants but starts with the current node
- `depth()` : calculate the depth of the current node (root node is 0)
- `path()` : gets the path to the current node

If you know the path to a group or layer within the tree, you can search by that path. Note that this always returns an Array because layer/group names do not have to be unique. The search is always scoped to the descendants of the current node, as well.

```
psd.tree().childrenAtPath('Version A/Matte');
psd.tree().childrenAtPath(['Version A', 'Matte']);
```

Accessing Layer Data

To get data such as the name or dimensions of a layer:

```
node = psd.tree().descendants()[0];
node.get('name');
node.get('width');
```

PSD files also store various pieces of information in "layer info" blocks. See this file for all of the possible layer info blocks that PSD.js parses (in `LAYER_INFO`). Which blocks a layer has varies from layer-to-layer, but to access them you can do:

```
node = psd.tree().descendants()[0]
node.get('typeTool').export()
node.get('vectorMask').export()
```

Exporting Data

When working with the tree structure, you can recursively export any node to an object. This does not dump *everything*, but it does include the most commonly accessed information.

```
console.log(psd.tree().export());
```

Which produces something like:

```
{ children:
   [ { type: 'group',
       visible: false,
       opacity: 1,
       blendingMode: 'normal',
       name: 'Version D',
       left: 0,
       right: 900,
       top: 0,
       bottom: 600,
       height: 600,
       width: 900,
       children:
        [ { type: 'layer',
            visible: true,
            opacity: 1,
            blendingMode: 'normal',
            name: 'Make a change and save.',
            left: 275,
            right: 636,
            top: 435,
            bottom: 466,
            height: 31,
            width: 361,
            mask: {},
            text:
             { value: 'Make a change and save.',
               font:
                { name: 'HelveticaNeue-Light',
                  sizes: [ 33 ],
                  colors: [ [ 85, 96, 110, 255 ] ],
                  alignment: [ 'center' ] },
               left: 0,
               top: 0,
               right: 0,
               bottom: 0,
               transform: { xx: 1, xy: 0, yx: 0, yy: 1, tx: 456, ty: 459 } },
            image: {} } ] } ],
    document:
       { width: 900,
         height: 600,
         resources:
          { layerComps:
             [ { id: 692243163, name: 'Version A', capturedInfo: 1 },
               { id: 725235304, name: 'Version B', capturedInfo: 1 },
               { id: 730932877, name: 'Version C', capturedInfo: 1 } ],
            guides: [],
            slices: [] } } }
```

You can also export the PSD to a flattened image. Please note that, at this time, not all image modes + depths are supported.

```
png = psd.image.toPng(); // get PNG object
psd.image.saveAsPng('path/to/output.png').then(function () {
  console.log('Exported!');
});
```

This uses the full rasterized preview provided by Photoshop. If the file was not saved with Compatibility Mode enabled, this will return an empty image.