

Front-end Components

por

Cristian Douce

Algunos términos en la práctica de front-end

- Coding styles
- Mejores prácticas (Best-practices)
- Estándares (Standars)
- etc..

Frameworks

- Backbone
- Angular
- Ember.js
- KnockOUT
- ... entre otros <<http://todomvc.com>>

Framework-agnostic

- jQuery / dom
- lodash / underscore
- socket.io
- ... muchos más

Components

“Los frameworks son geniales, pero no lo son todo.”

–Cristian Douce

“Los components son aquello que
agrega valor a tu producto.”

–Cristian Douce

Por qué es mejor pensar en components?

- Las cosas pequeñas son más fáciles de “mantener”
- “Extender” o “importar” pequeñas estructuras es más sencillo
- Con tareas más específicas se pueden escribir “tests” robustos y enfocados.

Qué define un componente front-end?

- Cualquier combinación de “assets” (javascript, css, imágenes, fonts)
- Con el objetivo de ser “reutilizado” o “compartido” entre una o más aplicaciones.

Qué tipos de componentes puedo construir?

Javascript

UI

UI + Javascript

Cómo?

- NPM + Browserify + others...
- Component v1 (github.com/componentjs/component)
- Duo (<http://duojs.org>)

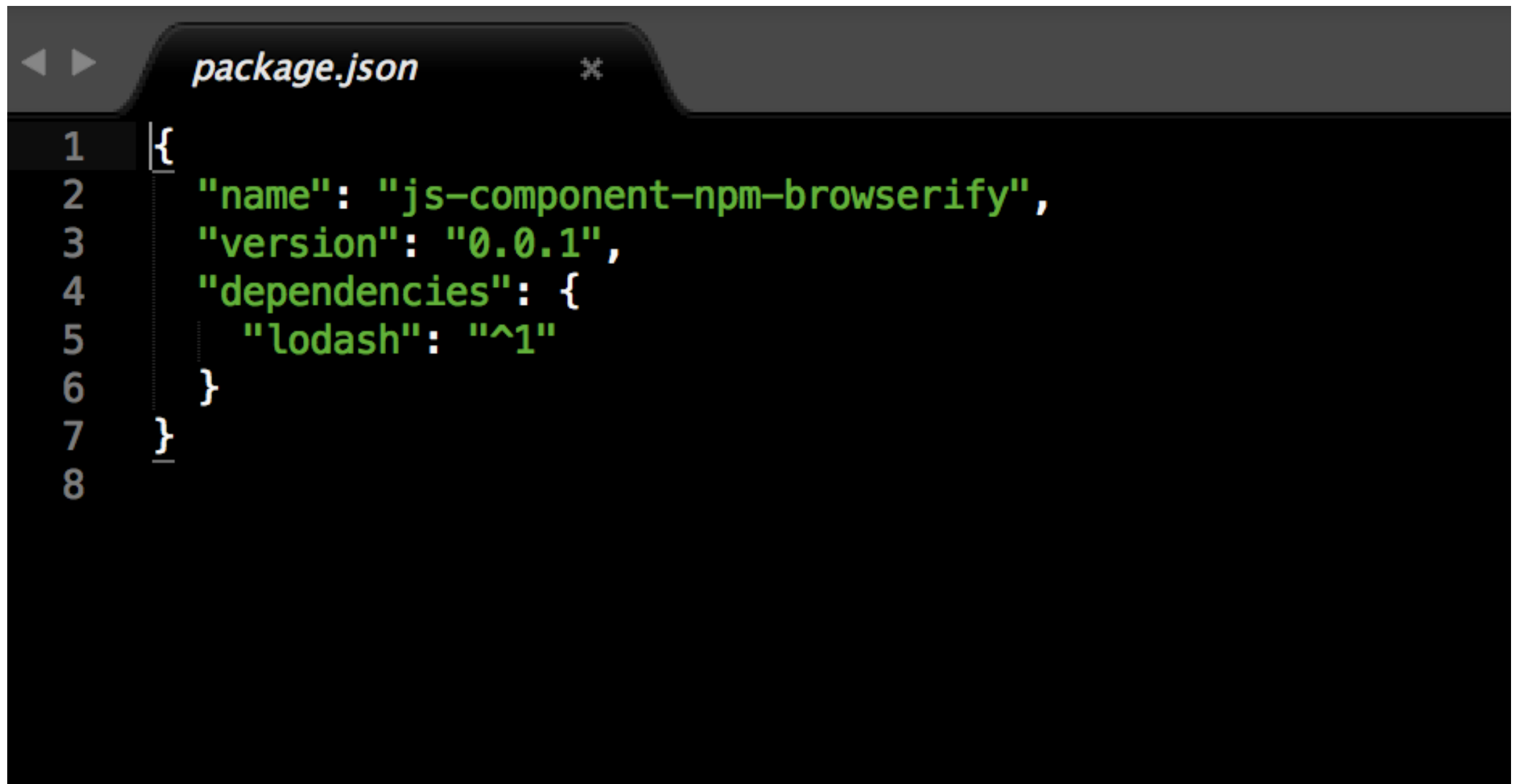
Componentes Javascript

Quiero un component que tome un array y me devuelva los números multiplicados por 3

- Instalar lodash como dependencia
- Escribir un index.js con el código
- Compilarlo con nuestra herramienta preferida
- Utilizarlo en un example.html

NPM + Browserify

npm install

A screenshot of a code editor with a dark theme. The editor has a tab at the top labeled 'package.json' with a close button (an asterisk). On the left side, there is a line number gutter with numbers 1 through 8. The code is written in a light green font. It is a JSON object representing a package configuration. The code is: { "name": "js-component-npm-browserify", "version": "0.0.1", "dependencies": { "lodash": "^1" } }

```
1 {  
2   "name": "js-component-npm-browserify",  
3   "version": "0.0.1",  
4   "dependencies": {  
5     "lodash": "^1"  
6   }  
7 }  
8
```

NPM + Browserify

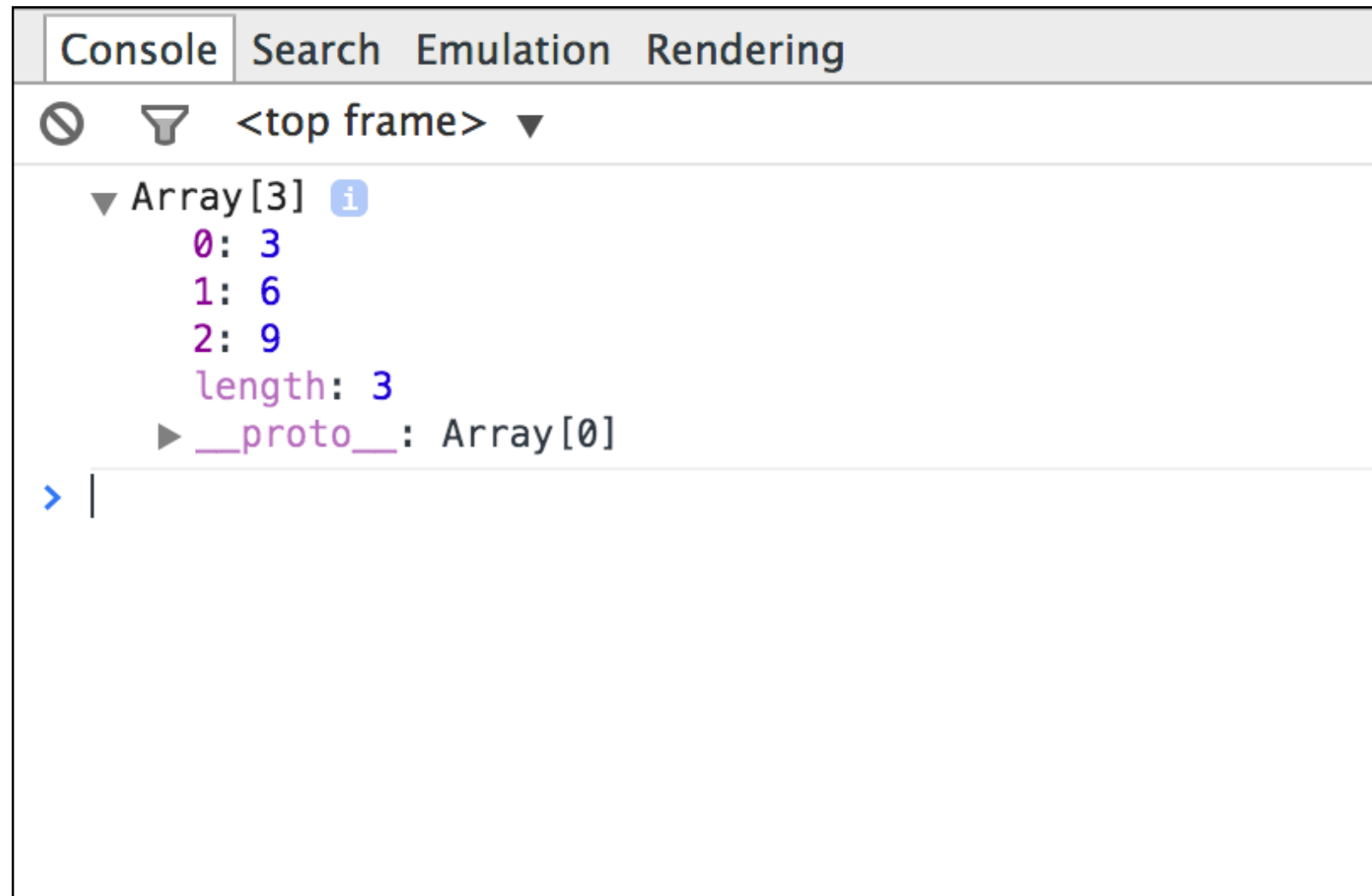
```
index.js
1  /**
2   * Module dependencies.
3   */
4
5  var _ = require('lodash');
6
7  /**
8   * Expose multiplyByThree
9   */
10
11  module.exports = multiplyByThree;
12
13  function multiplyByThree(arr) {
14    return _.map(arr, mutiplier);
15  }
16
17  function mutiplier(num) {
18    return num * 3;
19  }
20
```

NPM + Browserify

```
example.html *
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Javascript component - NPM + Browserify example</title>
5 </head>
6 <body>
7
8   <script type="text/javascript" src="build/build.js"></script>
9   <script type="text/javascript">
10     var multiplyByThree = require('multiply-by-three');
11
12     var result = multiplyByThree([1,2,3]);
13     console.log(result);
14   </script>
15 </body>
16 </html>
17
```

NPM + Browserify

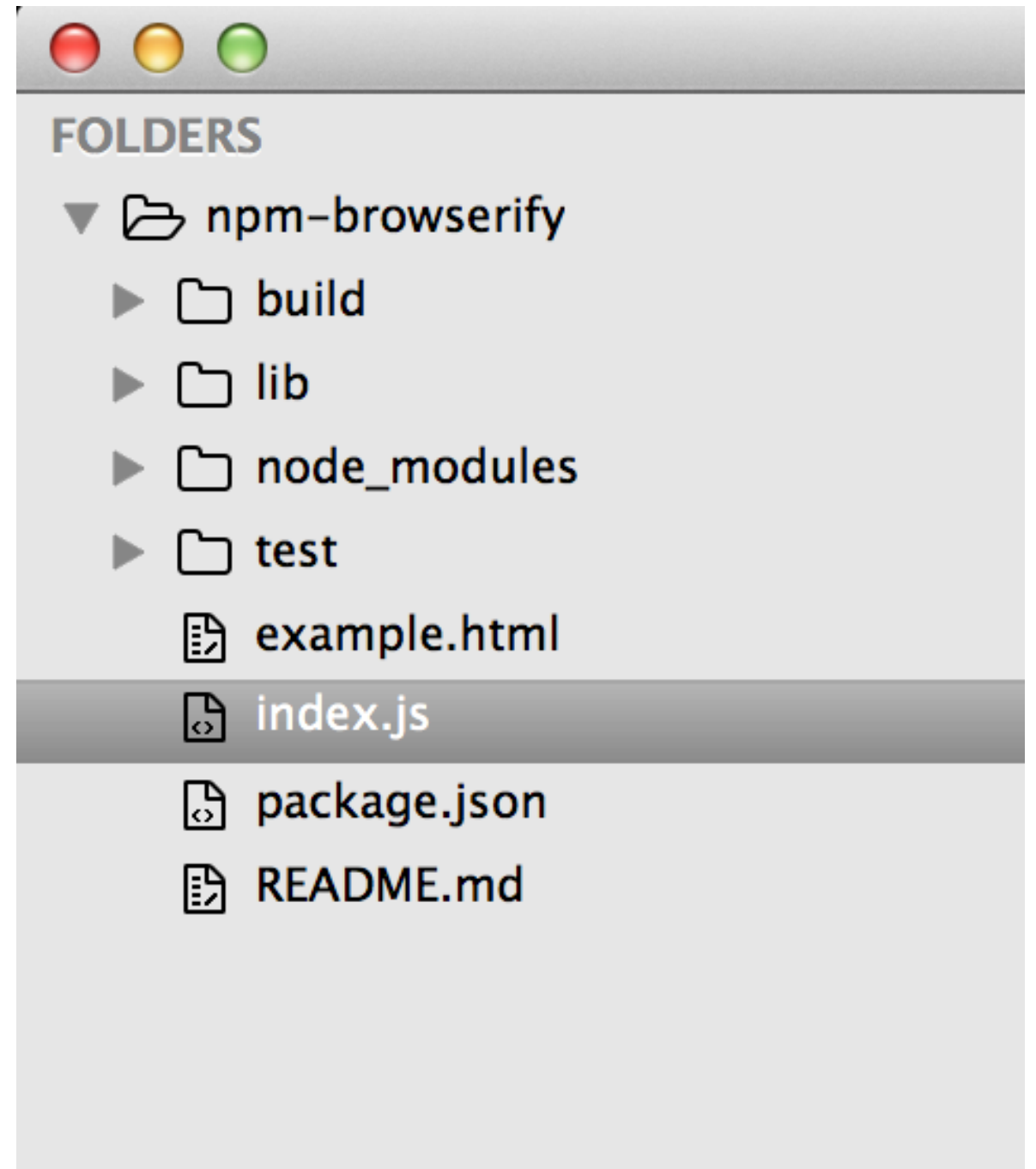
`browserify -r ./index.js:multiply-by-three -o build/build.css`



NPM + Browserify

Estructura de archivos

- Entry point - fuente
- test
- lib
- Meta-files



Component v1

component install

A screenshot of a code editor window with a dark theme. The title bar shows a file named 'component.json' with a close button. The editor displays a JSON configuration for a component. On the left side of the editor, line numbers 1 through 9 are visible. The JSON content is as follows:

```
1 {  
2   "name": "multiply-by-three",  
3   "version": "0.0.1",  
4   "dependencies": {  
5     "lodash/lodash": "^1"  
6   },  
7   "scripts": ["index.js"]  
8 }  
9
```

Component v1

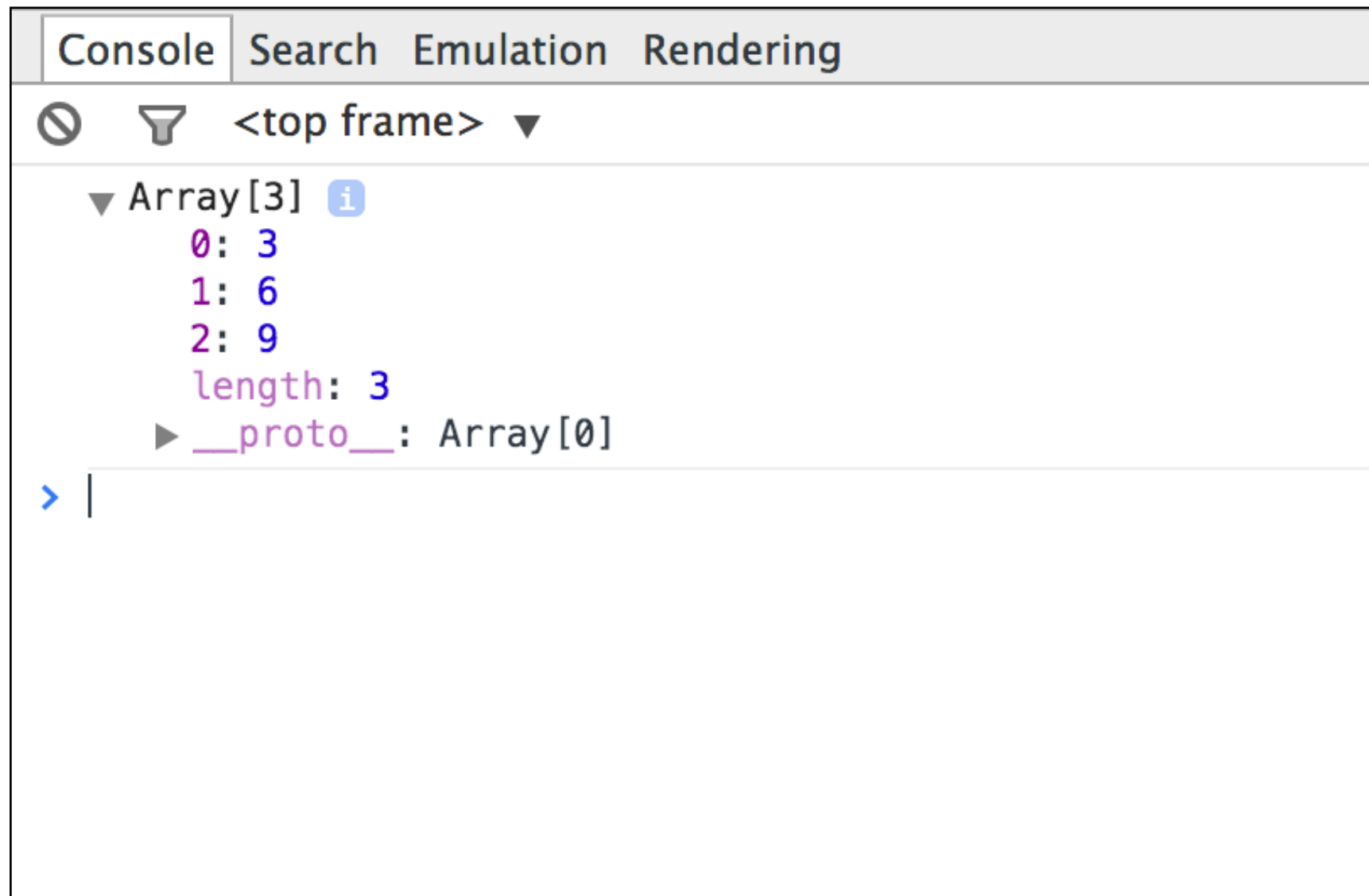
```
index.js
1  /**
2   * Module dependencies.
3   */
4
5  var _ = require('lodash');
6
7  /**
8   * Expose multiplyByThree
9   */
10
11 module.exports = multiplyByThree;
12
13 function multiplyByThree(arr) {
14   return _.map(arr, mutiplier);
15 }
16
17 function mutiplier(num) {
18   return num * 3;
19 }
20
```

Component v1

```
example.html *
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Javascript component - NPM + Browserify example</title>
5 </head>
6 <body>
7
8   <script type="text/javascript" src="build/build.js"></script>
9   <script type="text/javascript">
10     var multiplyByThree = require('multiply-by-three');
11
12     var result = multiplyByThree([1,2,3]);
13     console.log(result);
14   </script>
15 </body>
16 </html>
17
```

Component v1

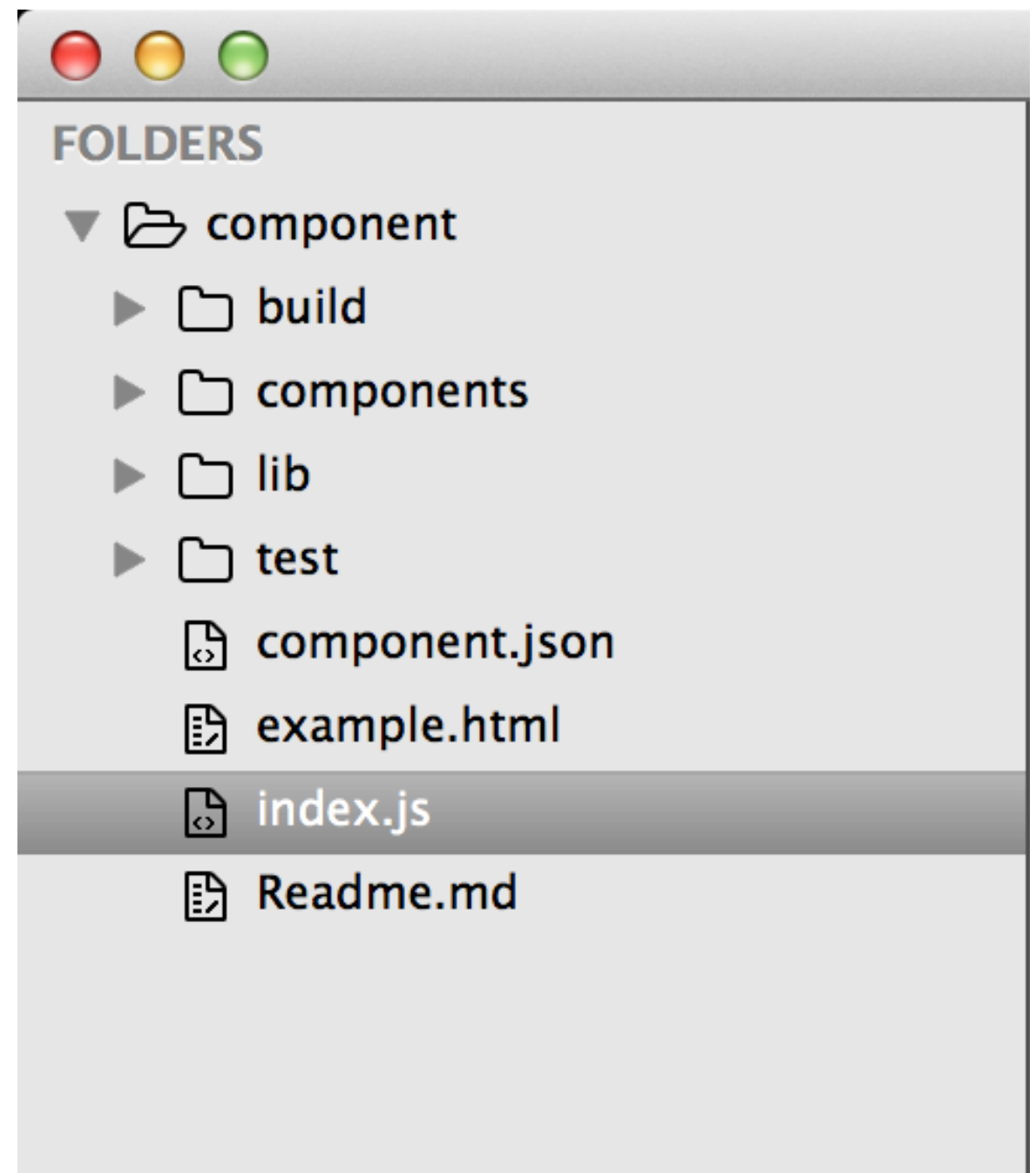
component build



Component v1

Estructura de archivos

- Entry point - fuente
- test
- lib
- Meta-files



Duo

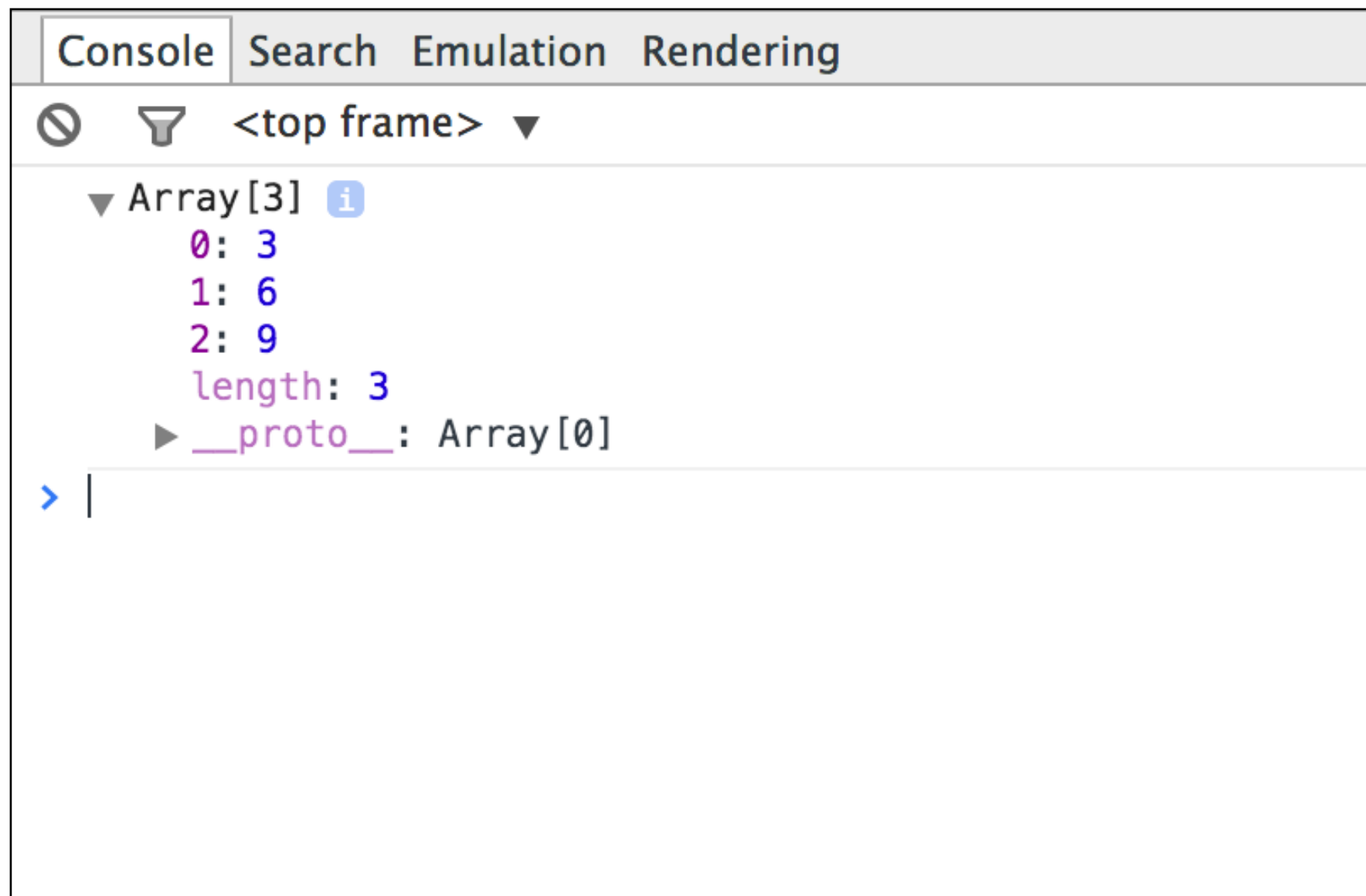
```
index.js
1  /**
2   * Module dependencies.
3   */
4
5  var _ = require('lodash/lodash');
6
7  /**
8   * Expose multiplyByThree
9   */
10
11  module.exports = multiplyByThree;
12
13  function multiplyByThree(arr) {
14    return _.map(arr, mutiplier);
15  }
16
17  function mutiplier(num) {
18    return num * 3;
19  }
20
```

Duo

```
example.html *
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Javascript component - Duo example</title>
5 </head>
6 <body>
7
8   <script type="text/javascript" src="build/build.js"></script>
9   <script type="text/javascript">
10     var result = multiplyByThree([1,2,3]);
11     console.log(result);
12   </script>
13 </body>
14 </html>
15
```


Duo

duo —global multiplyByThree index.js > build/build.js











Duo

Estructura de archivos

- Entry point - fuente
- test
- lib
- Meta-files

FOLDERS

- ▼  duo
 - ▶  build
 - ▶  components
 - ▶  lib
 - ▶  test
 -  example.html
 -  index.js
 -  Readme.md

Componentes de UI

Quiero un component que me ponga una imagen de fondo a cualquier elemento con la clase “bg-image” y además me haga reset de css del browser

- Instalar normalize.css como dependencia
- Escribir un index.css con el código
- Compilarlo con nuestra herramienta preferida
- Utilizarlo en un example.html

NPM + Browserify

No soporta CSS

npm install insert-css

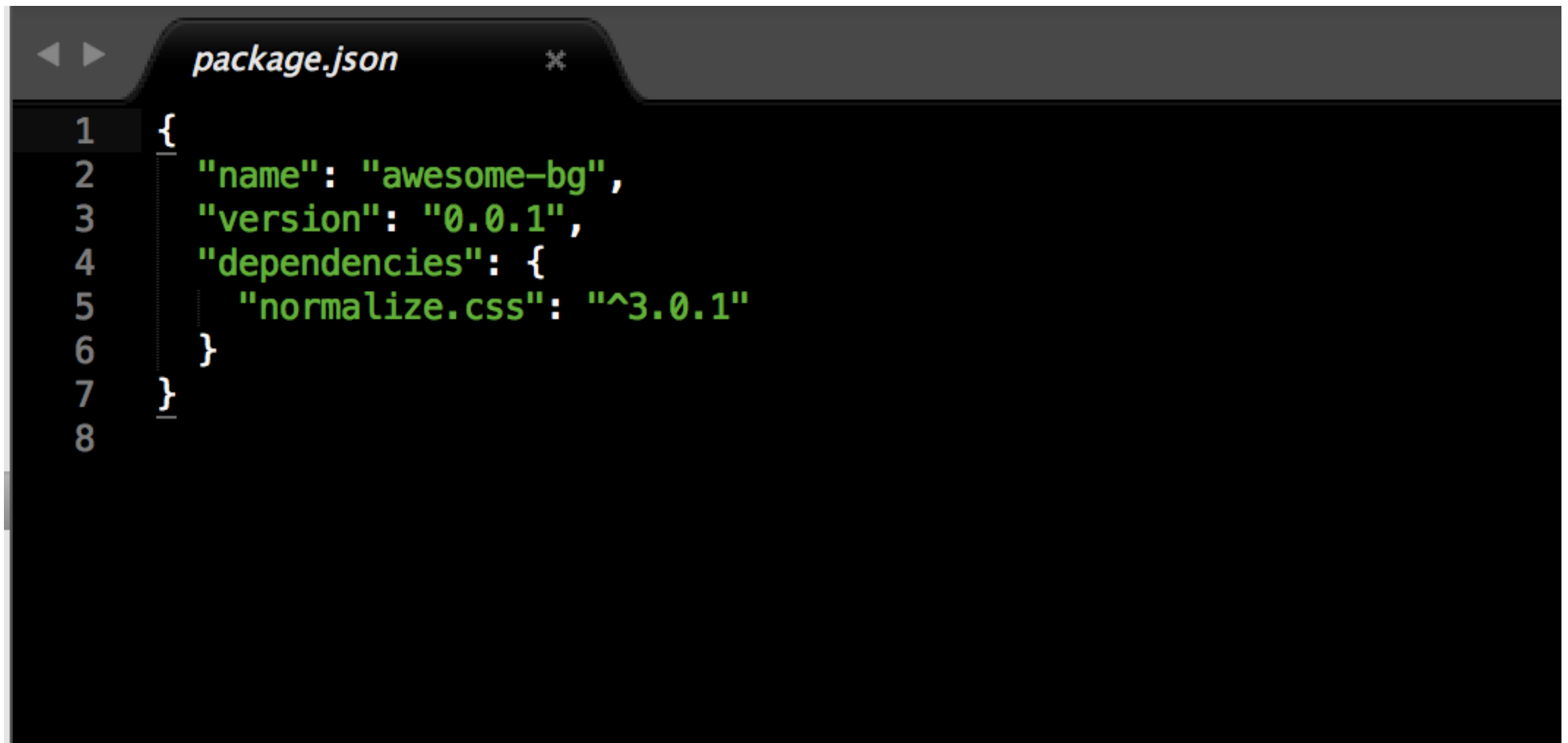
```
1  /**
2   * Module dependencies.
3   */
4
5  var insert = require('insert-css');
6  var CSS = require('fs').readFileSync('./index.css');
7
8  insert(CSS);
```

NPM + npm-css

npm install -g npm-css

NPM + npm-css

npm install

A screenshot of a code editor with a dark theme. The editor has a tab at the top labeled 'package.json' with a close button. On the left, there is a line number gutter with numbers 1 through 8. The code is written in a light green color on a black background. It is a JSON object representing a package configuration. The code is as follows:

```
1 {  
2   "name": "awesome-bg",  
3   "version": "0.0.1",  
4   "dependencies": {  
5     "normalize.css": "^3.0.1"  
6   }  
7 }  
8
```

NPM + npm-css

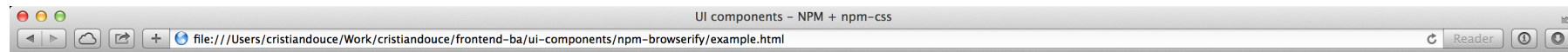
```
index.css *
1  @import "normalize.css";
2
3  .bg-image {
4      width: 400px;
5      height: 400px;
6      margin-top: 100px;
7      margin-left: auto;
8      margin-right: auto;
9      border: 2px solid #16214D;
10     background-image: url(../images/badge.png);
11     background-position: center center;
12     background-repeat: no-repeat;
13 }
14
```

NPM + npm-css

```
example.html *
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>UI components - NPM + npm-css</title>
5      <link rel="stylesheet" type="text/css" href="build.css">
6  </head>
7  <body>
8      <div class="bg-image"></div>
9  </body>
10 </html>
11 |
```


NPM + npm-css

`npm-css index.css -o build.css`













NPM + npm-css

Estructura de archivos

- Entry point - fuente
- fonts
- images
- Meta-files

FOLDERS

- ▼  npm-browserify
 - ▶  fonts
 - ▶  images
 - ▶  node_modules
 -  build.css
 -  example.html
 -  History.md
 -  index.css
 -  package.json
 -  Readme.md

Component v1

component install

A screenshot of a code editor window with a dark theme. The title bar shows a tab labeled 'component.json' with a close button. The editor contains a JSON file with the following content:

```
1  {
2    "name": "awesome-bg",
3    "version": "0.0.1",
4    "dependencies": {
5      "necolas/normalize.css": "^3.0.1"
6    },
7    "images": [
8      "images/badge.png"
9    ],
10   "styles": ["index.css"]
11 }
12
```

The line numbers 1 through 12 are visible on the left side of the editor. The JSON is formatted with green text on a dark background. The 'dependencies' object has one entry, 'necolas/normalize.css' with a version constraint of '^3.0.1'. The 'images' array contains one string, 'images/badge.png'. The 'styles' array contains one string, 'index.css'.

Component v1

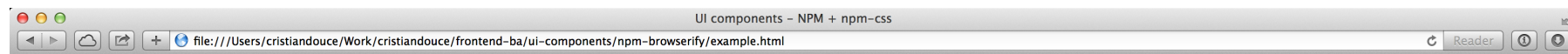
```
index.css
1
2 .bg-image {
3     width: 400px;
4     height: 400px;
5     margin-top: 100px;
6     margin-left: auto;
7     margin-right: auto;
8     border: 2px solid #16214D;
9     background-image: url(./images/badge.png);
10    background-position: center center;
11    background-repeat: no-repeat;
12 }
13
```

Component v1

```
example.html *
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>UI components - NPM + npm-css</title>
5   <link rel="stylesheet" type="text/css" href="build/build.css">
6 </head>
7 <body>
8   <div class="bg-image"></div>
9 </body>
10 </html>
11
```

Component v1

component build



Component v1

Estructura de archivos

- Entry point - fuente
- fonts
- images
- Meta-files

FOLDERS

- ▼ component
 - ▶ build
 - ▶ components
 - ▶ fonts
 - ▶ images
- component.json
- example.html
- History.md
- index.css
- Readme.md

Duo

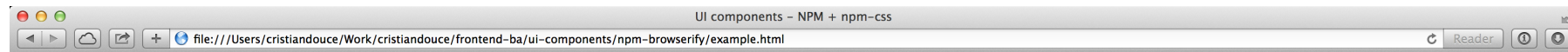
```
index.css
1  @import "necolas/normalize.css";
2
3  .bg-image {
4      width: 400px;
5      height: 400px;
6      margin-top: 100px;
7      margin-left: auto;
8      margin-right: auto;
9      border: 2px solid #16214D;
10     background-image: url(../images/badge.png);
11     background-position: center center;
12     background-repeat: no-repeat;
13 }
14
```


Duo

```
example.html *
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>UI components - NPM + npm-css</title>
5      <link rel="stylesheet" type="text/css" href="build/build.css">
6  </head>
7  <body>
8      <div class="bg-image"></div>
9  </body>
10 </html>
11
```

Duo

duo index.css > build/build.css



Duo

Estructura de archivos

- Entry point - fuente
- fonts
- images
- Meta-files

FOLDERS

- ▼ duo
 - ▶ build
 - ▶ components
 - ▶ fonts
 - ▶ images
 - example.html
 - History.md
 - index.css
 - Readme.md

Qué es mejor?

**Depende de la implementación
y
requerimientos del proyecto**

“Los components front-end son geniales!
Y si bien no lo son todo,
existe una tendencia que no hay que ignorar.”

–Cristian Douce

¡Gracias!



Cristian Douce



/cristiandouce



/cristiandouce