# Sensible Broccoli

# Sensible Broccoli

Covering the basics, and some real world use cases.

I'm a JavaScript Developer

# I'm a JavaScript Developer

And I've been writing JavaScript since...

# Before Crockford told us it was good.

*"Fortunately, JavaScript has some extraordinarily good parts. In JavaScript, there is a beautiful, elegant, highly expressive language...*

# Before Crockford told us it was good.

**And everyone else was still saying it was bad...**

*"Fortunately, JavaScript has some extraordinarily good parts. In JavaScript, there is a beautiful, elegant, highly expressive language...*

# Before Crockford told us it was good.

And everyone else was still saying it was bad... **Even him.**

*"Fortunately, JavaScript has some extraordinarily good parts. In JavaScript, there is a beautiful, elegant, highly expressive language that is buried under a steaming pile of good intentions and blunders."*

# We all were flash developers

# We all were flash developers

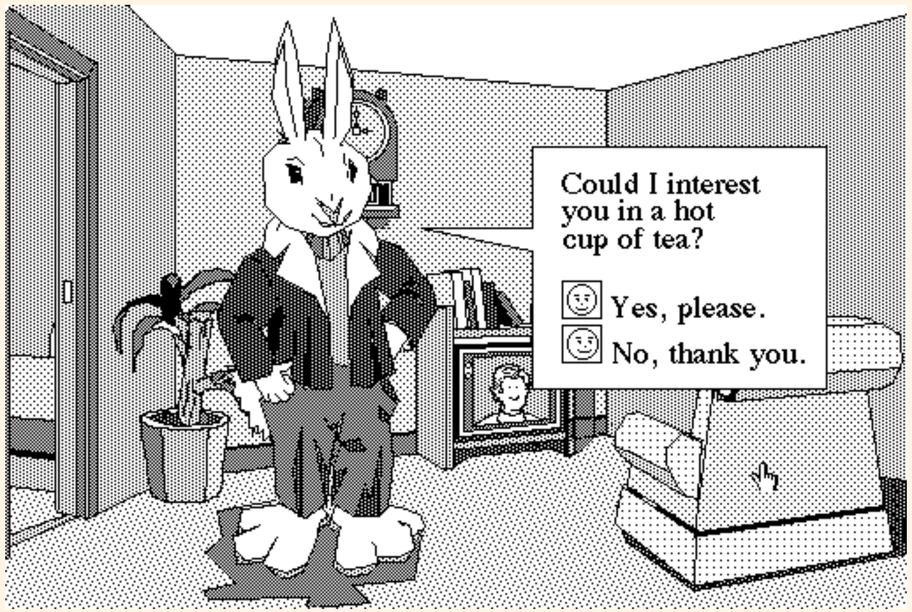
At least it wasn't shockwave.

# We all were flash developers

At least it wasn't shockwave. Or java applets.

# We all were flash developers

At least it wasn't shockwave. Or java applets. Or HyperCard..



# Yehuda was working for the enemy

*In Ruby, we have the great fortune to have one major framework (Rails) and a number of minor frameworks that drive innovation forward.*

# Yehuda was working for the enemy

**But there was still good in him.**

*"jQuery provides a powerful evented model that can be used to build up fairly complex applications*

# Even before node.js

# Even before node.js

Was in diapers...

# Was obtrusive and Contained HTML

```
render: function() {
    return (
     '<div>' +
       '<h3>Ask A Question</h3>' +
       '<p>' + exampleQuestion + '</p>' +
       '<form onSubmit="handleQuestion">' +
         '<input onChange="onChange" value="' + defaultQuestion + '" />' +
         '<button>Ask</button>' +
       '</form>' +
     '</div>'
    );
  }
```

# Was obtrusive and Contained HTML

Well... Some are still doing that one...

```
render: function() {
    return (
      <div>
        <h3>Ask A Question</h3>
        <p>{exampleQuestion}</p>
        <form onSubmit={handleQuestion}>
          <input onChange={onChange} value={defaultQuestion} />
          <button>Ask</button>
        </form>
      </div>
    );
  }
```

# Ember Talk Checklist..

# Ember Talk Checklist..

- [ ] Troll react.

# Ember Talk Checklist..

Troll react.

Anyway..

# Broccoli

There are two main aspects.

# There are two main aspects.

**Manipulating of file structures.**

# There are two main aspects.

Manipulating of file structures. **Manipulating of file contents.**

# Manipulating of File Structures

# Manipulating of File Structures

Bread and butter of broccoli.

# Manipulating of File Structures

Bread and butter of broccoli. The two main directives are..

# Funnels & Merges

# Basic Funnel of an App Directory

```
var tree = new Funnel('app');
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app');
```

```
└── app.js
└── components/
    ├── components/two-thing.js
├── index.html
├── resolver.js
└── router.js
└── routes/
    └── routes/one-thing/
        ├── routes/one-thing/component.js
        └── routes/one-thing/template.hbs
└── styles/
    └── styles/app.scss
└── templates/
    └── templates/components/
        └── templates/components/two-thing.hbs
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app', {
  srcDir: 'routes'
});
```

```
└── app.js
└── components/
    ├── components/two-thing.js
├── index.html
├── resolver.js
└── router.js
└── routes/
    └── routes/one-thing/
        ├── routes/one-thing/component.js
        └── routes/one-thing/template.hbs
└── styles/
    └── styles/app.scss
└── templates/
    └── templates/components/
        └── templates/components/two-thing.hbs
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app', {
  srcDir: 'routes'
});
```

```
└── one-thing/
    ├── one-thing/component.js
    └── one-thing/template.hbs
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app', {
  srcDir: 'routes',
  include: ['**/*.js']
});
```

```
└── one-thing/
    ├── one-thing/component.js
    └── one-thing/template.hbs
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app', {
  srcDir: 'routes',
  include: ['**/*.js']
});
```

```
└── one-thing/
    └── one-thing/component.js
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app');
```

```
└── app.js
└── components/
    ├── components/two-thing.js
├── index.html
├── resolver.js
└── router.js
└── routes/
    └── routes/one-thing/
        ├── routes/one-thing/component.js
        └── routes/one-thing/template.hbs
└── styles/
    └── styles/app.scss
└── templates/
    └── templates/components/
        └── templates/components/two-thing.hbs
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app', {
  srcDir: '.',
  include: ['**/*.hbs']
});
```

```
└── app.js
└── components/
    ├── components/two-thing.js
├── index.html
├── resolver.js
└── router.js
└── routes/
    └── routes/one-thing/
        ├── routes/one-thing/component.js
        └── routes/one-thing/template.hbs
└── styles/
    └── styles/app.scss
└── templates/
    └── templates/components/
        └── templates/components/two-thing.hbs
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app', {
  srcDir: '.',
  include: ['**/*.hbs']
});
```

```
└── routes/
    └── routes/one-thing/
        └── routes/one-thing/template.hbs
```

```
└── templates/
    └── templates/components/
        └── templates/components/two-thing.hbs
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app', {
  srcDir: '.',
  include: ['**/*.hbs'],
  exclude: ['templates/**']
});
```

```
└── routes/
    └── routes/one-thing/
        └── routes/one-thing/template.hbs
```

```
└── templates/
    └── templates/components/
        └── templates/components/two-thing.hbs
```

# Basic Funnel of an App Directory

```
var tree = new Funnel('app', {
  srcDir: '.',
  include: ['**/*.hbs'],
  exclude: ['templates/**']
});
```

```
└── routes/
    └── routes/one-thing/
        └── routes/one-thing/template.hbs
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');
```

```
└── app.js
└── components/
   ├── components/one-thing.js
├── index.html
├── resolver.js
└── router.js
└── routes/
   └── routes/three-thing/
      ├── routes/three-thing/component.js
      ├── routes/three-thing/route.js
      └── routes/three-thing/template.hbs
   └── routes/two-thing/
      ├── routes/two-thing/component.js
      └── routes/two-thing/template.hbs
└── styles/
   └── styles/app.css
└── templates/
   └── templates/components/
      └── templates/components/one-thing.hbs
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});
```

```
└── app.js
└── components/
    ├── components/one-thing.js
├── index.html
├── resolver.js
└── router.js
└── routes/
    └── routes/three-thing/
        ├── routes/three-thing/component.js
        ├── routes/three-thing/route.js
        └── routes/three-thing/template.hbs
    └── routes/two-thing/
        ├── routes/two-thing/component.js
        └── routes/two-thing/template.hbs
└── styles/
    └── styles/app.css
└── templates/
    └── templates/components/
        └── templates/components/one-thing.hbs
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});
```

```
└── three-thing/
    ├── three-thing/component.js
    ├── three-thing/route.js
    └── three-thing/template.hbs
└── two-thing/
    ├── two-thing/component.js
    └── two-thing/template.hbs
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});

var routeComponents = new Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components'
});
```

```
└── three-thing/
    ├── three-thing/component.js
    ├── three-thing/route.js
    └── three-thing/template.hbs
└── two-thing/
    ├── two-thing/component.js
    └── two-thing/template.hbs
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});

var routeComponents = new Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components'
});
```

```
└── components/
    └── components/three-thing/
        └── components/three-thing/component.js
    └── components/two-thing/
        └── components/two-thing/component.js
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});

var routeComponents = new Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component', '');
  }
});
```

```
└── components/
    └── components/three-thing/
        └── components/three-thing/component.js
    └── components/two-thing/
        └── components/two-thing/component.js
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});

var routeComponents = new Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component', '');
  }
});
```

```
└── components/
    ├── components/three-thing.js
    └── components/two-thing.js
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});

var routeComponents = new Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component', '');
  }
});

var routeTemplates = new Funnel(justRoutes, {
  include: ['**/template.hbs'],
  destDir: 'templates/components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/template', '');
  }
});
```

```
└── components/
    ├── components/three-thing.js
    └── components/two-thing.js
```

```
└── templates/
    └── templates/components/
        ├── templates/components/three-thing.hbs
        └── templates/components/two-thing.hbs
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});

var routeComponents = new Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component', '');
  }
});

var routeTemplates = new Funnel(justRoutes, {
  include: ['**/template.hbs'],
  destDir: 'templates/components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/template', '');
  }
});

var excludeRouteComponents = Funnel(appTree, {
  exclude: ['routes/**/{component.js,template.hbs}']
});
```

```
└── app.js
└── components/
   ├── components/one-thing.js
├── index.html
├── resolver.js
└── router.js
└── routes/
   └── routes/three-thing/
      ├── routes/three-thing/component.js
      ├── routes/three-thing/route.js
      └── routes/three-thing/template.hbs
   └── routes/two-thing/
      ├── routes/two-thing/component.js
      └── routes/two-thing/template.hbs
└── styles/
   └── styles/app.css
└── templates/
   └── templates/components/
      └── templates/components/one-thing.hbs
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});

var routeComponents = new Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component', '');
  }
});

var routeTemplates = new Funnel(justRoutes, {
  include: ['**/template.hbs'],
  destDir: 'templates/components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/template', '');
  }
});

var excludeRouteComponents = Funnel(appTree, {
  exclude: ['routes/**/{component.js,template.hbs}']
});
```

```
└── app.js
└── components/
   ├── components/one-thing.js
├── index.html
├── resolver.js
└── router.js
└── routes/
   └── routes/three-thing/

      ├── routes/three-thing/route.js

└── styles/
   └── styles/app.css
└── templates/
   └── templates/components/
      └── templates/components/one-thing.hbs
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});

var routeComponents = new Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component', '');
  }
});

var routeTemplates = new Funnel(justRoutes, {
  include: ['**/template.hbs'],
  destDir: 'templates/components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/template', '');
  }
});

var excludeRouteComponents = Funnel(appTree, {
  exclude: ['routes/**/{component.js,template.hbs}']
});

var alteredApp = new Merge([routeComponents, routeTemplates, excludeRouteComponents]);
```

```
└── components/
    ├── components/three-thing.js
    └── components/two-thing.js
```

```
└── templates/
    └── templates/components/
        ├── templates/components/three-thing.hbs
        └── templates/components/two-thing.hbs
```

# Putting funnels together and merging them.

```
var appTree = new Funnel('app');

var justRoutes = new Funnel(appTree, {
  srcDir: 'routes'
});

var routeComponents = new Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component', '');
  }
});

var routeTemplates = new Funnel(justRoutes, {
  include: ['**/template.hbs'],
  destDir: 'templates/components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/template', '');
  }
});

var excludeRouteComponents = Funnel(appTree, {
  exclude: ['routes/**/{component.js,template.hbs}']
});

var alteredApp = new Merge([routeComponents, routeTemplates, excludeRouteComponents]);
```

```
└── app.js
└── components/
    ├── components/one-thing.js
    ├── components/three-thing.js
    ├── components/two-thing.js
├── index.html
├── resolver.js
└── router.js
└── routes/
    └── routes/three-thing/
        └── routes/three-thing/route.js

└── styles/
    └── styles/app.css
└── templates/
    └── templates/components/
        ├── templates/components/one-thing.hbs
        ├── templates/components/three-thing.hbs
        └── templates/components/two-thing.hbs
```

Great. But now what?

# Great. But now what?

I didn't come here to just move files around.

# Great. But now what?

I didn't come here to just move files around. I want to manipulate there contents

# Adding Some Basic Filters

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});
```

# Adding Some Basic Filters

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});

routeComponents = new EsTranspiler(routeComponents);

routeComponents = new UglifyJS(routeComponents);
```

# What about debugging?

# Trying some debugging.

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});

routeComponents = new EsTranspiler(routeComponents);
console.log(routeComponents);
routeComponents = new UglifyJS(routeComponents);
```

```
Babel {
  _debug:
   { [Function: disabled]
     enabled: false,
     namespace: 'broccoli-persistent-filter:Babel' },
  _instantiationStack: '    at Babel.Plugin (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-babel-transpiler/node_modules/broccoli-persistent-filter/node_modules/broccoli-plugin/index.js:7:31)\n    at Babel.Filter [as constructor] (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-babel-transpiler/
node_modules/broccoli-persistent-filter/index.js:39:10)\n    at new Babel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-babel-transpiler/index.js:34:10)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-test/index.js:68:19)\n    at CoreObject._treeFor (/Users/mavery/Documents/
testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at
EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)',
  _name: 'Babel',
  _annotation: undefined,
  _baseConstructorCalled: true,
  _inputNodes:
   [ Funnel {
       _instantiationStack: '    at Funnel.Plugin (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/node_modules/broccoli-plugin/index.js:7:31)\n    at new Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:44:10)\n    at Funnel (/Users/mavery/Documents/testers/broccoli-example/
node_modules/broccoli-funnel/index.js:42:43)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-test/index.js:60:23)\n    at CoreObject._treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/
broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at
EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)',
       _name: 'Funnel',
       _annotation: undefined,
       _baseConstructorCalled: true,
       _inputNodes: [Object],
       _persistentOutput: true,
       _includeFileCache: {},
       _destinationPathCache: {},
       _currentTree: [Object],
       include: [Object],
       destDir: 'components',
       getDestinationPath: [Function: getDestinationPath],
       count: 0,
       _matchedWalk: true,
       _instantiatedStack: 'Error\n    at new Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:89:30)\n    at Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:42:43)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-
test/index.js:60:23)\n    at CoreObject._treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/
ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)\n
at EmberApp.appAndDependencies (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:1063:10)',
       _buildStart: undefined } ],
  _persistentOutput: true,
  processor:
   Processor {
     processor:
      { _cache: [Object],
        init: [Function],
        cacheKey: [Function],
        processString: [Function] },
     persistent: true },
  currentTree: FSTree { entries: [] },
  _counters:
   { hit: 0,
     prime: 0,
     patches: 0,
     operations: { mkdir: 0, rmdir: 0, unlink: 0, change: 0, create: 0, other: 0 },
     linked: 0,
     processed: 0 },
  _canProcessCache: {},
  _destFilePathCache: {},
  options: {},
  moduleMetadata: {},
  extensions: [ 'js' ],
  extensionsRegex: [ /.js$/ ],
  name: 'broccoli-babel-transpiler',
  inputTree:
   Funnel {
     _instantiationStack: '    at Funnel.Plugin (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/node_modules/broccoli-plugin/index.js:7:31)\n    at new Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:44:10)\n    at Funnel (/Users/mavery/Documents/testers/broccoli-example/
node_modules/broccoli-funnel/index.js:42:43)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-test/index.js:60:23)\n    at CoreObject._treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/
broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at
EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)',
     _name: 'Funnel',
     _annotation: undefined,
     _baseConstructorCalled: true,
     _inputNodes: [ [Object] ],
     _persistentOutput: true,
     _includeFileCache: {},
     _destinationPathCache: {},
     _currentTree: FSTree { entries: [] },
     include: [ [Object] ],
     destDir: 'components',
     getDestinationPath: [Function: getDestinationPath],
     count: 0,
     _matchedWalk: true,
     _instantiatedStack: 'Error\n    at new Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:89:30)\n    at Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:42:43)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-
test/index.js:60:23)\n    at CoreObject._treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/
ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)\n
at EmberApp.appAndDependencies (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:1063:10)',
     _buildStart: undefined } }
```

```
Babel {
  _debug:
   { [Function: disabled]
     enabled: false,
     namespace: 'broccoli-persistent-filter:Babel' },
  _instantiationStack: '    at Babel.Plugin (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-babel-transpiler/node_modules/broccoli-persistent-filter/node_modules/broccoli-plugin/index.js:7:31)\n    at Babel.Filter [as constructor] (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-babel-transpiler/
node_modules/broccoli-persistent-filter/index.js:39:10)\n    at new Babel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-babel-transpiler/index.js:34:10)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-test/index.js:68:19)\n    at CoreObject._treeFor (/Users/mavery/Documents/
testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at
EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)',
  _name: 'Babel',
  _annotation: undefined,
  _baseConstructorCalled: true,
  _inputNodes:
   [ Funnel {
       _instantiationStack: '    at Funnel.Plugin (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/node_modules/broccoli-plugin/index.js:7:31)\n    at new Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:44:10)\n    at Funnel (/Users/mavery/Documents/testers/broccoli-example/
node_modules/broccoli-funnel/index.js:42:43)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-test/index.js:60:23)\n    at CoreObject._treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/
broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at
EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)',
       _name: 'Funnel',
       _annotation: undefined,
       _baseConstructorCalled: true,
       _inputNodes: [Object],
       _persistentOutput: true,
       _includeFileCache: {},
       _destinationPathCache: {},
       _currentTree: [Object],
       include: [Object],
       destDir: 'components',
       getDestinationPath: [Function: getDestinationPath],
       count: 0,
       _matchedWalk: true,
       _instantiatedStack: 'Error\n    at new Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:89:30)\n    at Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:42:43)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-
test/index.js:60:23)\n    at CoreObject._treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/
ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)\n
at EmberApp.appAndDependencies (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:1063:10)',
       _buildStart: undefined } ],
  _persistentOutput: true,
  processor:
   Processor {
     processor:
      { _cache: [Object],
        init: [Function],
        cacheKey: [Function],
        processString: [Function] },
     persistent: true },
  currentTree: FSTree { entries: [] },
  _counters:
   { hit: 0,
     prime: 0,
     patches: 0,
     operations: { mkdir: 0, rmdir: 0, unlink: 0, change: 0, create: 0, other: 0 },
     linked: 0,
     processed: 0 },
  _canProcessCache: {},
  _destFilePathCache: {},
  options: {},
  moduleMetadata: {},
  extensions: [ 'js' ],
  extensionsRegex: [ /.js$/ ],
  name: 'broccoli-babel-transpiler',
  inputTree:
   Funnel {
     _instantiationStack: '    at Funnel.Plugin (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/node_modules/broccoli-plugin/index.js:7:31)\n    at new Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:44:10)\n    at Funnel (/Users/mavery/Documents/testers/broccoli-example/
node_modules/broccoli-funnel/index.js:42:43)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-test/index.js:60:23)\n    at CoreObject._treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/
broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at
EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)',
     _name: 'Funnel',
     _annotation: undefined,
     _baseConstructorCalled: true,
     _inputNodes: [ [Object] ],
     _persistentOutput: true,
     _includeFileCache: {},
     _destinationPathCache: {},
     _currentTree: FSTree { entries: [] },
     include: [ [Object] ],
     destDir: 'components',
     getDestinationPath: [Function: getDestinationPath],
     count: 0,
     _matchedWalk: true,
     _instantiatedStack: 'Error\n    at new Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:89:30)\n    at Funnel (/Users/mavery/Documents/testers/broccoli-example/node_modules/broccoli-funnel/index.js:42:43)\n    at CoreObject.treeForApp (/Users/mavery/Documents/testers/broccoli-example/lib/broccoli-
test/index.js:60:23)\n    at CoreObject._treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:373:33)\n    at CoreObject.treeFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/models/addon.js:341:21)\n    at /Users/mavery/Documents/testers/broccoli-example/node_modules/
ember-cli/lib/broccoli/ember-app.js:498:20\n    at Array.map (native)\n    at EmberApp.addonTreesFor (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:496:30)\n    at EmberApp._processedAppTree (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:792:25)\n
at EmberApp.appAndDependencies (/Users/mavery/Documents/testers/broccoli-example/node_modules/ember-cli/lib/broccoli/ember-app.js:1063:10)',
     _buildStart: undefined } }
```


# Not what we where looking for

# Trying some debugging.

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});

routeComponents = new EsTranspiler(routeComponents);
console.log(fs.readFileSync('app/components/three-thing.js', 'utf8'));
routeComponents = new UglifyJS(routeComponents);
```

ENOENT: no such file or directory, open 'app/components/three-thing.js'

# Trying some debugging.

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});

routeComponents = new EsTranspiler(routeComponents);
console.log(fs.readFileSync('app/routes/three-thing/component.js', 'utf8'));
routeComponents = new UglifyJS(routeComponents);
```

```
import Ember from 'ember';

export default Ember.Component.extend({
});
```

Still not what we where after.

# Trying some debugging.

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});

routeComponents = new EsTranspiler(routeComponents);
console.log(routeComponents);
console.log(fs.readFileSync('app/components/three-thing.js', 'utf8'));
console.log(fs.readFileSync('app/routes/three-thing/component.js', 'utf8'));
routeComponents = new UglifyJS(routeComponents);
```

There are also two main phases.

There are also two main phases.

**Setup and Build.**

# Setup

# Setup

So this is what we have been doing so far.

# These are all run in the setup phase.

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});

routeComponents = new EsTranspiler(routeComponents);

routeComponents = new UglifyJS(routeComponents);
```

Back to the debugging example.

# Back to the debugging example.

And enter `'broccoli-stew'`

# Back to the debugging example.

And enter `'broccoli-stew'` And lets keep the debugging in the build phase.

# Keeps the build debugging in the build.

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});

routeComponents = new EsTranspiler(routeComponents);

routeComponents = new UglifyJS(routeComponents);
```

# Keeps the build debugging in the build.

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});

routeComponents = new EsTranspiler(routeComponents);
console.log(routeComponents);
routeComponents = new UglifyJS(routeComponents);
```

# Keeps the build debugging in the build.

```
var routeComponents = Funnel(justRoutes, {
  include: ['**/component.js'],
  destDir: 'components',
  getDestinationPath(relativePath) {
    return relativePath.replace('/component.js', '.js');
  }
});

routeComponents = new EsTranspiler(routeComponents);
routeComponents = Stew.log(routeComponents, {output: 'tree'});
routeComponents = new UglifyJS(routeComponents);
```

# Keeps the build debugging in the build.

```
routeComponents = new EsTranspiler(routeComponents);
routeComponents = Stew.log(routeComponents, {output: 'tree'});
```

```
└── components/
    ├── components/three-thing.js
    └── components/two-thing.js
```

# Keeps the build debugging in the build.

```
routeComponents = new EsTranspiler(routeComponents);
routeComponents = Stew.map(routeComponents, function(content, relativePath) {
  console.log(content);
  return content;
});
```

# Keeps the build debugging in the build.

```
routeComponents = new EsTranspiler(routeComponents);
routeComponents = Stew.map(routeComponents, function(content, relativePath) {
  console.log(content);
  return content;
});
```

```
'use strict';

Object.defineProperty(exports, '__esModule', {
  value: true
});

function _interopRequireDefault(obj) { return obj && obj.__esModule ? obj : { 'default': obj }; }

var _ember = require('ember');

var _ember2 = _interopRequireDefault(_ember);

exports['default'] = _ember2['default'].Component.extend({});
module.exports = exports['default'];
```

And this is the power of the build.

# And this is the power of the build.

Cause you only setup once,

# And this is the power of the build.

Cause you only setup once, **but you build every time.**

Lets unpack this build a little more

# Lets unpack this build a little more

Using on of the build plugins `'broccoli-filter'`

# Basic Filter for uglify

```
var Filter = require('broccoli-filter');
var UglifyJS = require('uglify-js');

module.exports = UglifyJSFilter;
UglifyJSFilter.prototype = Object.create(Filter.prototype);
UglifyJSFilter.prototype.constructor = UglifyJSFilter;
function UglifyJSFilter (inputTree, options) {
  Filter.call(this, inputTree, options);
  this.options = options || {};
};

UglifyJSFilter.prototype.extensions = ['js'];
UglifyJSFilter.prototype.targetExtension = 'js';

UglifyJSFilter.prototype.processString = function (string, relativePath) {
  var result = UglifyJS.minify(string, {
    ...
  })
  return result.code;
};
```

# Basic Filter for uglify

```
var Filter = require('broccoli-filter');
var UglifyJS = require('uglify-js');

module.exports = UglifyJSFilter;
UglifyJSFilter.prototype = Object.create(Filter.prototype);
UglifyJSFilter.prototype.constructor = UglifyJSFilter;
function UglifyJSFilter (inputTree, options) {
  Filter.call(this, inputTree, options);
  this.options = options || {};
};

UglifyJSFilter.prototype.extensions = ['js'];
UglifyJSFilter.prototype.targetExtension = 'js';

UglifyJSFilter.prototype.processString = function (string, relativePath) {
  var result = UglifyJS.minify(string, {
    ...
  })
  return result.code;
};
```

# Basic Filter for uglify

```
var Filter = require('broccoli-filter');
var UglifyJS = require('uglify-js');

module.exports = UglifyJSFilter;
UglifyJSFilter.prototype = Object.create(Filter.prototype);
UglifyJSFilter.prototype.constructor = UglifyJSFilter;
function UglifyJSFilter (inputTree, options) {
  Filter.call(this, inputTree, options);
  this.options = options || {};
};

UglifyJSFilter.prototype.extensions = ['js'];
UglifyJSFilter.prototype.targetExtension = 'js';

UglifyJSFilter.prototype.processString = function (string, relativePath) {
  var result = UglifyJS.minify(string, {
    ...
  })
  return result.code;
};
```

# Basic Filter for uglify

```
var Filter = require('broccoli-filter');
var UglifyJS = require('uglify-js');

module.exports = UglifyJSFilter;
UglifyJSFilter.prototype = Object.create(Filter.prototype);
UglifyJSFilter.prototype.constructor = UglifyJSFilter;
function UglifyJSFilter (inputTree, options) {
  Filter.call(this, inputTree, options);
  this.options = options || {};
};

UglifyJSFilter.prototype.extensions = ['js'];
UglifyJSFilter.prototype.targetExtension = 'js';

UglifyJSFilter.prototype.processString = function (string, relativePath) {
  var result = UglifyJS.minify(string, {
    ...
  })
  return result.code;
};
```

# Basic Filter for uglify

```
var Filter = require('broccoli-filter');
var UglifyJS = require('uglify-js');

module.exports = UglifyJSFilter;
UglifyJSFilter.prototype = Object.create(Filter.prototype);
UglifyJSFilter.prototype.constructor = UglifyJSFilter;
function UglifyJSFilter (inputTree, options) {
  Filter.call(this, inputTree, options);
  this.options = options || {};
};

UglifyJSFilter.prototype.extensions = ['js'];
UglifyJSFilter.prototype.targetExtension = 'js';

UglifyJSFilter.prototype.processString = function (string, relativePath) {
  var result = UglifyJS.minify(string, {
    ...
  })
  return result.code;
};
```

# Basic Filter for uglify

```
var Filter = require('broccoli-filter');
var UglifyJS = require('uglify-js');

module.exports = UglifyJSFilter;
UglifyJSFilter.prototype = Object.create(Filter.prototype);
UglifyJSFilter.prototype.constructor = UglifyJSFilter;
function UglifyJSFilter (inputTree, options) {
  Filter.call(this, inputTree, options);
  this.options = options || {};
};

UglifyJSFilter.prototype.extensions = ['js'];
UglifyJSFilter.prototype.targetExtension = 'js';

UglifyJSFilter.prototype.processString = function (string, relativePath) {
  var result = UglifyJS.minify(string, {
    ...
  })
  return result.code;
};
```

# Basic Filter for uglify

```
var Filter = require('broccoli-filter');
var UglifyJS = require('uglify-js');

module.exports = UglifyJSFilter;
UglifyJSFilter.prototype = Object.create(Filter.prototype);
UglifyJSFilter.prototype.constructor = UglifyJSFilter;
function UglifyJSFilter (inputTree, options) {
  Filter.call(this, inputTree, options);
  this.options = options || {};
};

UglifyJSFilter.prototype.extensions = ['js'];
UglifyJSFilter.prototype.targetExtension = 'js';

UglifyJSFilter.prototype.processString = function (string, relativePath) {
  var result = UglifyJS.minify(string, {
    ...
  })
  return result.code;
};
```

# Basic Filter for uglify

```
var Filter = require('broccoli-filter');
var UglifyJS = require('uglify-js');

module.exports = UglifyJSFilter;
UglifyJSFilter.prototype = Object.create(Filter.prototype);
UglifyJSFilter.prototype.constructor = UglifyJSFilter;
function UglifyJSFilter (inputTree, options) {
  Filter.call(this, inputTree, options);
  this.options = options || {};
};

UglifyJSFilter.prototype.extensions = ['js'];
UglifyJSFilter.prototype.targetExtension = 'js';

UglifyJSFilter.prototype.processString = function (string, relativePath) {
  var result = UglifyJS.minify(string, {
    ...
  })
  return result.code;
};
```

# Basic Filter for uglify

```
var Filter = require('broccoli-filter');
var UglifyJS = require('uglify-js');

module.exports = UglifyJSFilter;
UglifyJSFilter.prototype = Object.create(Filter.prototype);
UglifyJSFilter.prototype.constructor = UglifyJSFilter;
function UglifyJSFilter (inputTree, options) {
  Filter.call(this, inputTree, options);
  this.options = options || {};
};

UglifyJSFilter.prototype.extensions = ['js'];
UglifyJSFilter.prototype.targetExtension = 'js';

UglifyJSFilter.prototype.processString = function (string, relativePath) {
  var result = UglifyJS.minify(string, {
    ...
  })
  return result.code;
};
```

So that's the basics of broccoli.

# So that's the basics of broccoli.

Now for some real world applications.

'ember-component-css'

'ember-component-css'

This addon revolutionized the way I write css.

'ember-component-css'

This addon revolutionized the way I write css. But it was in an old crufty state

# Styles are contextual to you component.

```
└── app.js
└── index.html
└── pod-component/
    ├── pod-component/component.js
    ├── pod-component/styles.scss
    ├── pod-component/template.hbs
├── resolver.js
└── router.js
└── styles/
    └── styles/app.css
```

```scss
& {
  display: flex;
}
h2 {
  margin-top: 0;
  color: $midnight;
}
h5 {
  margin: 0;
  a {
    &:hover {
      text-decoration: underline;
    }
    &:before {
      @include line-triangle;
      border-color: $medium;
    }
  }
}
```

# Styles are contextual to you component.

```
└── app.js
└── index.html
└── pod-component/
    ├── pod-component/component.js
    ├── pod-component/styles.scss
    ├── pod-component/template.hbs
├── resolver.js
└── router.js
└── styles/
    └── styles/app.css
```

```scss
.__checklist-group__0c8ee {
  display: flex;
}
.__checklist-group__0c8ee h2 {
  margin-top: 0;
  color: $midnight;
}
.__checklist-group__0c8ee h5 {
  margin: 0;
  a {
    &:hover {
      text-decoration: underline;
    }
    &:before {
      @include line-triangle;
      border-color: $medium;
    }
  }
}
```

So lets unpack some of the pieces.

# So lets unpack some of the pieces.

First, the filter that makes this change.

# Filter for adding on class names.

```
treeForStyles: function(tree) {
  var podStyles = this._getPodStyleFunnel();

  podStyles = new PodStyles(podStyles, {
    extensions: this.allowedStyleExtensions,
    annotation: 'Filter (ember-component-css process component with class names)'
  });

  var styleManifest = new IncludeAll(podStyles, {
    annotation: 'IncludeAll (ember-component-css combining all style files that there are extensions for)'
  });

  podStyles = new Merge([podStyles, styleManifest, tree].filter(Boolean), {
    annotation: 'Merge (ember-component-css merge namespacedStyles with style manafest)'
  });

  return this._super.treeForStyles.call(this, podStyles);
},
```

# Filter for adding on class names.

```
treeForStyles: function(tree) {
  var podStyles = this._getPodStyleFunnel();

  podStyles = new PodStyles(podStyles, {
    extensions: this.allowedStyleExtensions,
    annotation: 'Filter (ember-component-css process component with class names)'
  });

  var styleManifest = new IncludeAll(podStyles, {
    annotation: 'IncludeAll (ember-component-css combining all style files that there are extensions for)'
  });

  podStyles = new Merge([podStyles, styleManifest, tree].filter(Boolean), {
    annotation: 'Merge (ember-component-css merge namespacedStyles with style manafest)'
  });

  return this._super.treeForStyles.call(this, podStyles);
},
```

# Filter for adding on class names.

```
function PodStyles(inputTree, options) {
  options = options || {};
  Filter.call(this, inputTree, {
    name: 'pod-styles-add-name',
    annotation: options.annotation
  });
  this.extensions = options.extensions;
};

PodStyles.prototype.processString = function(contents, stylePath) {
  var extension = path.extname(stylePath),
      className = componentNames.class(stylePath);
      strategy = 'default';

  switch (extension) {
    case '.styl':
    case '.sass':
      strategy = 'indentation';
      break;
    case '.less':
    case '.scss':
      strategy = 'syntax';
      break;
  }

  return processStratagies[strategy](contents, className, extension);
};
```

# Filter for adding on class names.

```
function PodStyles(inputTree, options) {
  options = options || {};
  Filter.call(this, inputTree, {
    name: 'pod-styles-add-name',
    annotation: options.annotation
  });
  this.extensions = options.extensions;
};

PodStyles.prototype.processString = function(contents, stylePath) {
  var extension = path.extname(stylePath),
      className = componentNames.class(stylePath);
      strategy = 'default';

  switch (extension) {
    case '.styl':
    case '.sass':
      strategy = 'indentation';
      break;
    case '.less':
    case '.scss':
      strategy = 'syntax';
      break;
  }

  return processStratagies[strategy](contents, className, extension);
};
```

# Filter for adding on class names.

```
function PodStyles(inputTree, options) {
  options = options || {};
  Filter.call(this, inputTree, {
    name: 'pod-styles-add-name',
    annotation: options.annotation
  });
  this.extensions = options.extensions;
};

PodStyles.prototype.processString = function(contents, stylePath) {
  var extension = path.extname(stylePath),
      className = componentNames.class(stylePath);
      strategy = 'default';

  switch (extension) {
    case '.styl':
    case '.sass':
      strategy = 'indentation';
      break;
    case '.less':
    case '.scss':
      strategy = 'syntax';
      break;
  }

  return processStratagies[strategy](contents, className, extension);
};
```

So that's a basic filter you might see.

# So that's a basic filter you might see.

Now for something a little more...

# So that's a basic filter you might see.

Now for something a little more... Tricky

# Finding the class names to use for the component.

```
PodNames.prototype.build = function() {
  var srcDir = this.inputPaths[0];

  var entries = walkSync.entries(srcDir);
  var nextTree = new FSTree.fromEntries(entries, { sortAndExpand: true });
  var currentTree = this.currentTree;

  this.currentTree = nextTree;
  var patches = currentTree.calculatePatch(nextTree);

  return Promise.resolve().then(this.writePodStyleName.bind(this, patches));
};

PodNames.prototype.writePodStyleName = function(patches) {
  for (var i = 0; i < patches.length; i++) {
    switch (patches[i][0]) {
      case 'create':
        this.addClass(patches[i][1]);
        break;
      case 'unlink':
        this.removeClass(patches[i][1]);
        break;
    }
  }

  var output = 'export default ' + JSON.stringify(this.podNameJson);
  return fs.writeFileSync(path.join(this.outputPath, 'pod-names.js'), output);
}
```