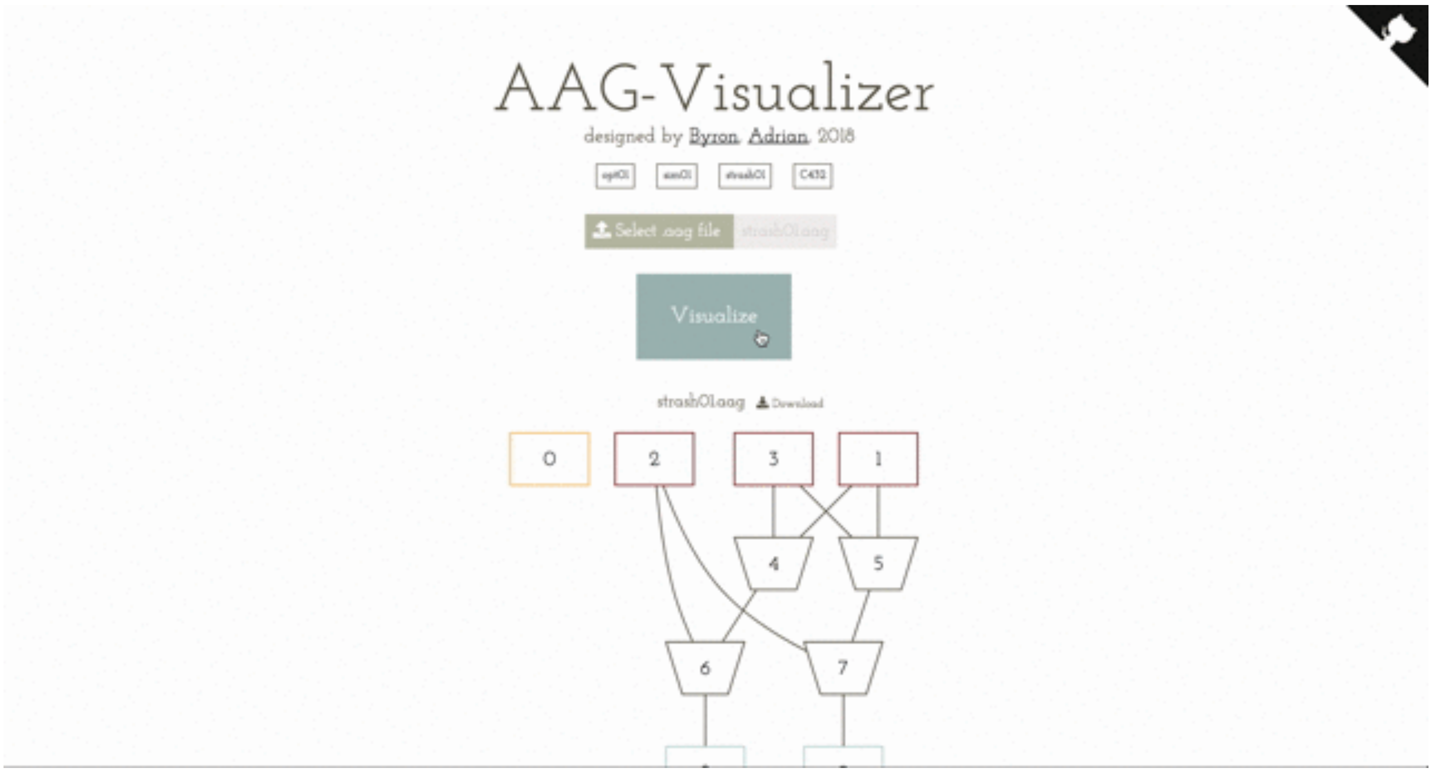
Visualize Any Graph with Pure Javascript !

> A complete guide to build your own visualizer using viz.js and animate.css.
> This tutorial is not specific to visualize circuit, it can apply to any graph you want.

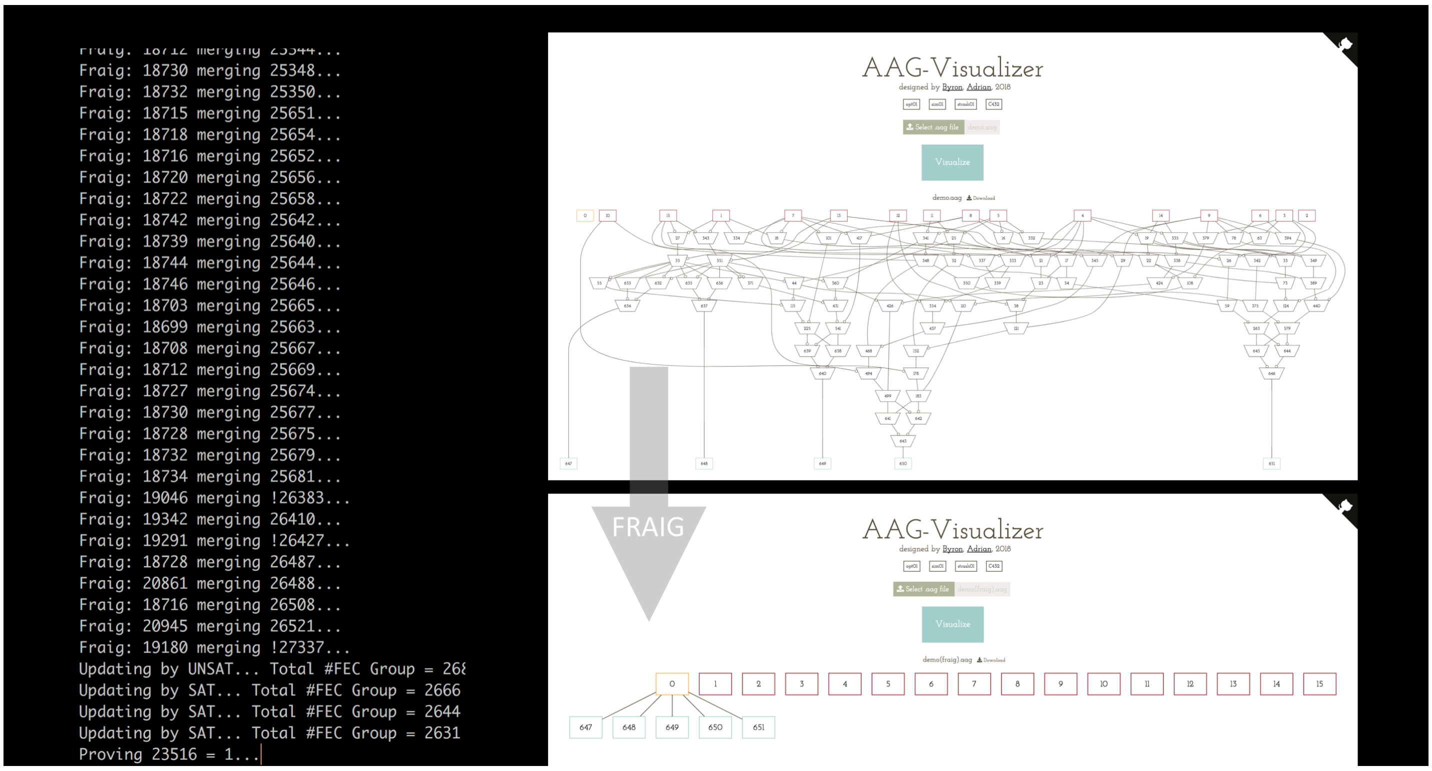


What is AAG-Visualizer? What can it do?
It is an elegant visualizer which can convert complex circuit format like AIGER (hard to understand to human) into simple and pretty svg with just one click.

> Inspration
> Final project【Fraig】of **Data Structure and Programming** (held by Ric Huang) In National Taiwan University Electrical Engineering Department

A brief introduction of Fraig and AAG

What does FRAIG do?



```
Fraig: 18712 merging 25344...
Fraig: 18730 merging 25348...
Fraig: 18732 merging 25350...
Fraig: 18715 merging 25651...
Fraig: 18718 merging 25654...
Fraig: 18716 merging 25652...
Fraig: 18720 merging 25656...
Fraig: 18722 merging 25658...
Fraig: 18742 merging 25642...
Fraig: 18739 merging 25640...
Fraig: 18744 merging 25644...
Fraig: 18746 merging 25646...
Fraig: 18703 merging 25665...
Fraig: 18699 merging 25663...
Fraig: 18708 merging 25667...
Fraig: 18712 merging 25669...
Fraig: 18727 merging 25674...
Fraig: 18730 merging 25677...
Fraig: 18728 merging 25675...
Fraig: 18732 merging 25679...
Fraig: 18734 merging 25681...
Fraig: 19046 merging !26383...
Fraig: 19342 merging 26410...
Fraig: 19291 merging !26427...
Fraig: 18728 merging 26487...
Fraig: 20861 merging 26488...
Fraig: 18716 merging 26508...
Fraig: 20945 merging 26521...
Fraig: 19180 merging !27337...
Updating by UNSAT... Total #FEC Group = 26
Updating by SAT... Total #FEC Group = 2666
Updating by SAT... Total #FEC Group = 2644
Updating by SAT... Total #FEC Group = 2631
Proving 23516 = 1...
```

Functionally Reducing And-Inverter Graph

What is AAG?

Check Brief Guide of Aiger Formot by my friend, Hanmo Ou.

You don't need to understand it thoroughly. What you need to know is that it is just a format of And-Invertor Graph.

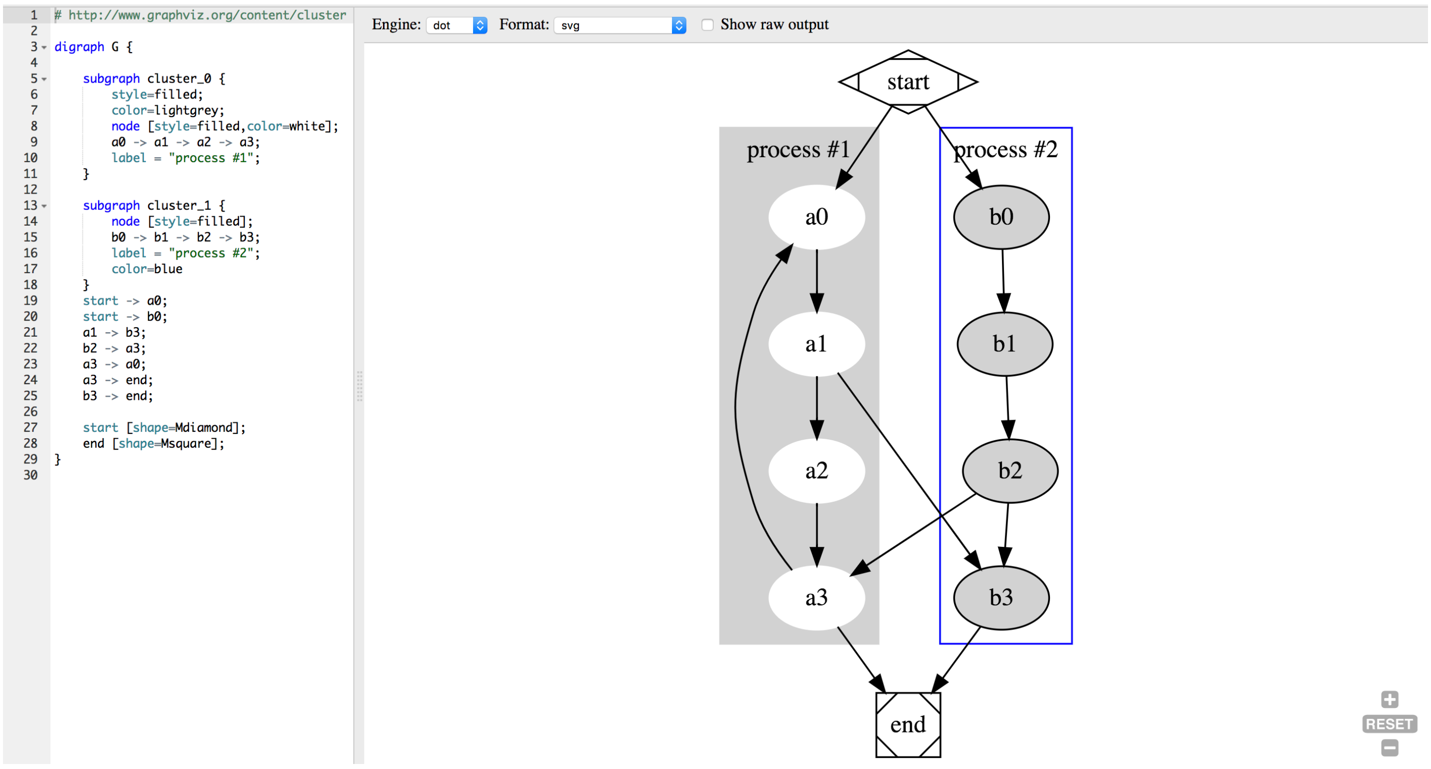
> **Ok, so let's get into the topic. How did I convert aag to svg on the frontend side?**

Application Structure

```
.
├── .gitignore
├── README.md
├── package.json
├── public
|    ├── bundle.js
|    ├── index.html
|    └── assets/
├── server
|    ├── config.js
|    └── server.js
├── src
|    ├── assets
|    ├── js
|    |    └── animate.js
|    |    └── parser.js
|    ├── scss
|    |    └──  index.scss
|    ├── index.js
|    └── template.html
├── webpack.dev.config.js
└── webpack.prod.config.js
```

Viz.js

official docs



```
# http://www.graphviz.org/content/cluster

digraph G {

    subgraph cluster_0 {
        style=filled;
        color=lightgrey;
        node [style=filled,color=white];
        a0 -> a1 -> a2 -> a3;
        label = "process #1";
    }

    subgraph cluster_1 {
        node [style=filled];
        b0 -> b1 -> b2 -> b3;
        label = "process #2";
        color=blue
    }
    start -> a0;
    start -> b0;
    a1 -> b3;
    b2 -> a3;
    a3 -> a0;
    a3 -> end;
    b3 -> end;

    start [shape=Mdiamond];
    end [shape=Msquare];
}
```

Engine: dot Format: svg Show raw output

Brief Introduction of Viz.js

This project is a Makefile for building Graphviz with Emscripten and a simple wrapper for using it in the browser. A super fast and convenient way to draw graphs on the browser.

- Install

```
npm install viz.js
```

- Basic Usage in javascript

```
// format of .dot
var digraph = 'digraph { a -> b; }';

//for svg
var svgXml = Viz(digraph, { format: "svg"});
document.body.innerHtml = svgXml;

//for img-element
var img-element = Viz(digraph, { format: "png-image-element"});
document.body.append(img-element);
```

- Customization

What's more? You can customize the node color, the vertex width...etc on the graph.
Just roll up your sleeves and change the style at your will.
See the docs here.

Nothing else! It is just that easy.
Now we only need to parse our input into .dot file.
And we can see our string magically turns into pretty svg circuit!

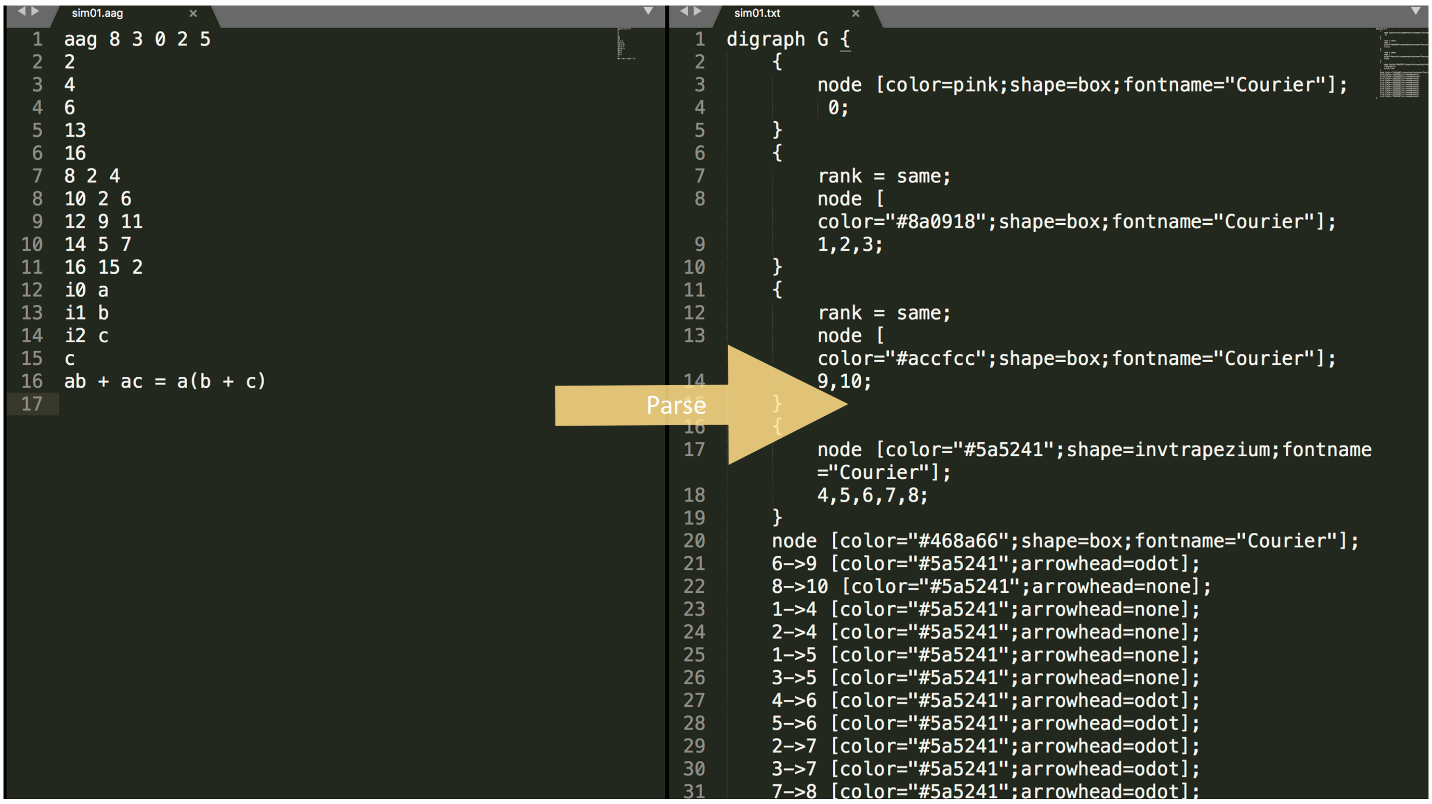
Parser

- parser.js

Use the mightiness of regexp in javascript.You can parse anything with ease.
Check out this fantastic tutorial!
JavaScript: Learn Regular Expressions for Beginners

> Best js regexp tutorial I have ever seen.

In view of different input format should have their own parser. I skip the part of discussing how to parse them. But I believe that if you can use regexp well. It is really not a big deal.



```
aag 8 3 0 2 5
2
4
6
13
16
8 2 4
10 2 6
12 9 11
14 5 7
16 15 2
i0 a
i1 b
i2 c
c
ab + ac = a(b + c)
```

```
digraph G {
    {
        node [color=pink;shape=box;fontname="Courier"];
        0;
    }
    {
        rank = same;
        node [
        color="#8a0918";shape=box;fontname="Courier"];
        1,2,3;
    }
    {
        rank = same;
        node [
        color="#accfcc";shape=box;fontname="Courier"];
        9,10;
    }
    {
        node [color="#5a5241";shape=invtrapezium;fontname
        ="Courier"];
        4,5,6,7,8;
    }
    node [color="#468a66";shape=box;fontname="Courier"];
    6->9 [color="#5a5241";arrowhead=odot];
    8->10 [color="#5a5241";arrowhead=none];
    1->4 [color="#5a5241";arrowhead=none];
    2->4 [color="#5a5241";arrowhead=none];
    1->5 [color="#5a5241";arrowhead=none];
    3->5 [color="#5a5241";arrowhead=none];
    4->6 [color="#5a5241";arrowhead=odot];
    5->6 [color="#5a5241";arrowhead=odot];
    2->7 [color="#5a5241";arrowhead=odot];
    3->7 [color="#5a5241";arrowhead=odot];
    7->8 [color="#5a5241";arrowhead=odot];
```

```
//parser.js
function parser(args...){
}
export default parser;
```

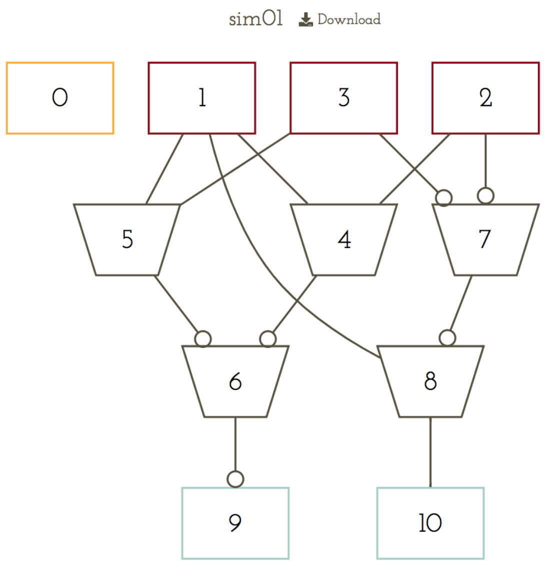
Combine

```
import parser from './js/parser.js'

// format of .dot
var digraph = parser(data);

//for svg
var svgXml = Viz(digraph, { format: "svg"});
document.body.innerHtml = svgXml;

//for img-element
var img-element = Viz(digraph, { format: format: "png-image-element"});
document.body.append(img-element);
```



Enjoy the fantastic fancy graph you make!

animate.css

[official docs]

Brief Introduction of animate.css

A super easy library to make fantastic animation on dom object.

- Install

```
<head>
  <!-- animate.css -->
  <link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.5.2/animate.min.css
  <!-- jquery -->
  <script
    src="https://code.jquery.com/jquery-3.1.1.min.js"
    integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
    crossorigin="anonymous"></script>
</head>
```

- Basic Usage

```
// animate.js
$.fn.extend({
    animateCss: function (animationName, callback) {
        var animationEnd = 'webkitAnimationEnd mozAnimationEnd MSAnimationEnd oanima
        this.addClass('animated ' + animationName).one(animationEnd, function() {
            $(this).removeClass('animated ' + animationName);
            if (callback) {
              callback();
            }
        });
        return this;
    }
});

// index.js
// load animate.js in advance for repeated animation
import './js/animate.js'
document.getElementById('container').animateCss('shake');
```

Implement

> My goal is to add "shake" animation on visualize button when the format is wrong.
> And add "squeeze" animation on it when it is visualized successfully.

How to do?

It is unbelievably easy!
You don't need to write a lot of keyframes and annoying thing.
Just call `animateCss` and the magic happen.

- Visualize Button

```
// index.js
function handleFileSelected(event) {
    ...
    if(input.files.length === 0) { // no file in <input>
        $('#btn').animateCss('shake');
        return;
    }else
    if(!/.aag/g.test(input.files[0].name)){ // format is wrong
        $('#btn').animateCss('shake');
        return;
    }
    ...
    // Right format and ready to visualize
    $('#btn').animateCss('rubberBand');
}
document.getElementById('btn').addEventListener('click', handleFileSelected);
```

img

Write 10 lines of code but impress all the user who step into your website.
Why not?