# Noteworks – A Better Way to Take Notes

By: Liam Doran (2014), Advisor: Professor Ridgway
Trinity College Computer Science Department

## Elements 2

- **Node**: Topic, focus of attention
- **Note**: Brief, single piece of information
- **Link:** Connection between two nodes
- **Category**: Marker of similar nodes

## Markup 3

**Goals:**
- Parse-able: Comprehensible to backend
- Simple: New users can learn quickly
- Robust: Organizational options for user

**Node:** *.Title*
**Node & Category:** *.Category Name, Title*
**Notes:** *-Content*
**Links:** *:Title, Title, Title, etc.*

## Introduction 1

### Objective
Build a web application that uses graph drawing and a simple interface to help users create cleaner, better organized, and more usable notes.

### Motivation
Studies have shown that notes taken on a computer and with organization in mind result in better learning. But, most notes today are barely organized and physical, making them hard to maintain and study from.

### Approach/Design
The application works by allowing a user to type in notes with a simple markup language I designed. The app parses these notes, breaks them down on the backend, and on the frontend it presents the user with an interactive graph of their data that updates in real time as they type their notes. Noteworks was built with Ruby on Rails, Cytoscape.js, and Foundation, as well as JavaScript, Ruby, Haml, and SASS.

## Data Flow 6

### Initialization

**Backend(Ruby)**
- Retrieves notes from database.
  - Graph components parsed to JSON.
  - Structural components parsed to text
    - Text inserted into the markup

**Frontend(JavaScript)**
- Receives JSON, initiates graph drawing.
  - Elements populated on canvas, arranged by force-directed algorithm.

### Live Updating

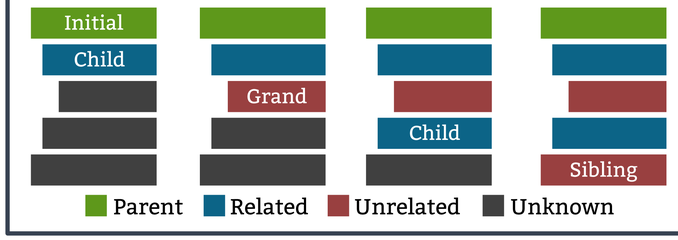
**Backend(Ruby)**
- Submitted text parsed, changes determined.
  - Database (data and relations) updated
  - Changes tracked (ARM), converted to JSON

**Frontend(JavaScript)**
- Watches text for changes in caret position, number of lines, selection size, and content.
  - If criteria met, change type (ARM) determined, AJAX call with line text & numbers.
- AJAX response captured, graph updated with submitted JSON.

## Parser 4

### Text
- Splits strings by type markers ( **.** , **-** , or **:** )
  - First character (marker) determines type
    - If not a valid marker, classified as null
- Left side: amount of whitespace determines depth
- Right side: content. Broken down depending on type.

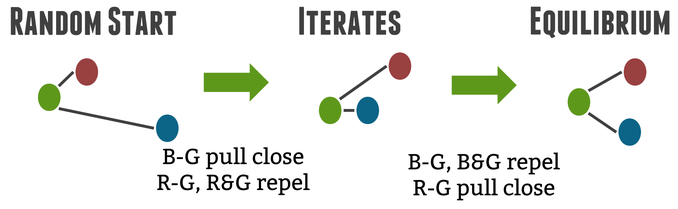
### Element Relations
- All elements can have a parent
- Only nodes can have children
- Algorithm begins at a node and steps through elements
  - If node found → its depth becomes child depth
  - If curr_depth == child depth → relation made (child)
  - If curr_depth > child depth → ignored (grandchild)
  - If curr_depth == initial node depth → ends (sibling)




## Graphing 5

### Algorithm
Uses a **Force-Directed Graph Drawing Algorithm**
- Models the graph as a physical system
  - Nodes repel each other
  - Edges are springs, resist compression and stretching

The graph iterates until it reaches an equilibrium, where the benefits of further changes fall below a threshold.




### Implementation
Force-directed implementation: Arbor.js by Samizdat Drafting
Graphing library: Cytoscape.js by Max Franz
- Canvas Renderer
  - Modified to allow for line wrapping
- Node and Graph Manipulation (clicking, dragging)

## References 7

Bui, Dung C., Joel Myerson, and Sandra Hale. "Note-taking with Computers: Exploring Alternative Strategies for Improved Recall." *Journal of Educational Psychology*105.2 (2013): 299-309. *American Psychological Association*. Web.

"Cytoscape.js Documentation." *Cytoscape.github.io.* April-17-2014. Web. March-10-2014.

Hua, Jie, Mao L. Huang, Weidong Huang, Junhu Wang, and Quang V. Nguyen. "Force-directed Graph Visualization with Pre-positioning - Improving Convergence Time and Quality of Layout." *16th International Conference on Information Visualization* (2012): 124-29. Web. <http://www.computer.org/csdl/proceeedings/iv/2012...>